IMT School for Advanced Studies, Lucca
Lucca, Italy

Optimizing complex networks models

PhD Program in Systems Science

Track in Economics, Networks and Business Analytics

XXXIII Cycle

By

Emiliano Marchese

2022

The dissertation of Emiliano Marchese is approved.

PhD Program Coordinator: Prof. Rocco De Nicola, IMT School for Advanced Studies Lucca

Advisor: Prof. Guido Caldarelli, "Ca' Foscari" University of Venice

Co-Advisor: Prof. Tiziano Squartini, IMT School for Advanced Studies Lucca

The dissertation of Emiliano Marchese has been reviewed by:

Prof. Angelo Bifone, University of Turin

Prof. Giacomo Livan, University College London

IMT School for Advanced Studies Lucca
2022

Contents

List of Figures

List of Tables

Acknowledgements

Each chapter of the present thesis is based on a co-authored paper (either published or still under submission) that is cited in the corresponding abstract. The colleagues with whom I have collaborated and that I would like to thank are Guido Caldarelli, Giulio Cimini, Diego Garlaschelli, Jian-Hong Lin, Fabio Saracco, Tiziano Squartini, Claudio Tessone, Giuseppe Trapani, Nicolò Vallarano and Mario Zanon.

Vita

January 4, 1990	Born in Catania (Italy)
2013	Bachelor Degree in Physics Final mark: 105/110 University of Catania (Catania, Italy)
2017	Master Degree in Physics Final mark: 110/110 University of Catania (Catania, Italy)
2017-2022	PhD in Systems Science (track in Economics, Networks and Business Analytics) at the IMT School for Advanced Studies Lucca (Lucca, Italy)
2019	Visiting period, within the Erasmus program, at Leiden University (Leiden, The Netherlands)
2019	Visiting period, within the Erasmus program, at ABN AMRO Bank (Amsterdam, The Netherlands)

Publications

1. **Spectral properties of random networks**
   EMILIANO MARCHESE, VALENTINA MACCHIATI, PIERO MAZZARISI, DIEGO GARLASCHELLI, TIZIANO SQUARTINI
   in preparation (2022)

2. **Structural analysis of the Mexican interbank network**
   EMILIANO MARCHESE, DROR KENNET, SERAFIN MARTINEZ-JARAMILLO, TIZIANO SQUARTINI
   in preparation (2022)

3. **The weighted Bitcoin Lightning Network**
   JIAN-HONG LIN, EMILIANO MARCHESE, CLAUDIO J. TESSONE, TIZIANO SQUARTINI
   arxiv:2111.13494 (submitted to Chaos, Solitons & Fractals) (2022)

4. **Reconstructing firm-level interactions in the Dutch input-output network from production constraints**
   LEONARDO NICCOLÒ IALONGO, CAMILLE DE VALK, EMILIANO MARCHESE, FABIAN JANSEN, HICHAM ZMARROU, TIZIANO SQUARTINI, DIEGO GARLASCHELLI
   arxiv:2111.15248 (accepted for publication on Scientific Reports) (2022)

5. **Detecting mesoscale structures by surprise**
   EMILIANO MARCHESE, GUIDO CALDARELLI, TIZIANO SQUARTINI
   Communications Physics 5 (132) (2022)

6. **Fast and scalable likelihood maximization for Exponential Random Graph Models with local constraints**
   NICOLÒ VALLARANO, MATTEO BRUNO, EMILIANO MARCHESE, GIUSEPPE TRAPANI, FABIO SARACCO, GIULIO CIMINI, MARIO ZANON, TIZIANO SQUARTINI
   Scientific Reports 11 (15227) (2021)

Presentations

1. **Networks 2021 - A Joint Sunbelt and NetSci Conference (online, 2021)**
   Detecting mesoscale structures by surprise
   EMILIANO MARCHESE, TIZIANO SQUARTINI
2. **Netsci 2020 - Network Science Society Conference (Rome, Italy, 2020)**
   Spectral signals of topological collapse in financial networks
   EMILIANO MARCHESE, TIZIANO SQUARTINI, GUIDO CALDARELLI
3. **Simulation-Based Science Club (IAS, Amsterdam, The Netherlands, 2019)**
   Spectral approach to stability analysis of financial networks
   EMILIANO MARCHESE, TIZIANO SQUARTINI

Posters

1. **Netsci 2020 - Network Science Society Conference (Rome, Italy, 2020)**
   Detecting mesoscale structures by surprise
   EMILIANO MARCHESE, TIZIANO SQUARTINI
2. **Big Data and Economic Forecasting Workshop (JRC, Ispra, Italy, 2019)**
   Spectral approach to stability analysis of financial networks
   EMILIANO MARCHESE, TIZIANO SQUARTINI

Codes

1. **NEMtropy: Network Entropy Maximization, a Toolbox Running On PYthon**
   GitHub repository - https://github.com/nicoloval/NEMtropy
   MATTEO BRUNO, EMILIANO MARCHESE, NICOLÒ VALLARANO
2. **SurpriseMeMore**
   GitHub repository - https://github.com/EmilianoMarchese/SurpriseMeMore
   EMILIANO MARCHESE

Press coverage

1. **SoBigData Newsletter**
   Detecting mesoscale structures by surprise
   EMILIANO MARCHESE, TIZIANO SQUARTINI

Abstract

Analyzing real-world networks ultimately amounts at comparing their empirical properties with the outcome of a proper, statistical model. The far most common, and most useful, approach to define benchmarks rests upon the so-called *canonical formalism* of statistical mechanics which has led to the definition of the broad class of models known as *Exponential Random Graphs* (ERGs). Generally speaking, employing a model of this family boils down at maximizing a likelihood function that embodies the available information about a certain system, hence constituting the desired benchmark. Although powerful, the aforementioned models cannot be solved analytically, whence the need to rest upon numerical recipes for their optimization. Generally speaking, this is a hard task, since real-world networks can be enormous in size (for example, consisting of billions of nodes and links), hence requiring models with 'many' parameters (say, of the same order of magnitude of the number of nodes). This evidence calls for optimization algorithms which are both fast and scalable: the collection of works constituting the present thesis represents an attempt to fill this gap. Chapter 1 provides a quick introduction to the topic. Chapter 2 deals specifically with ERGs: after reviewing the basic concepts constituting the pillars upon which such a framework is based, we will discuss several instances of it and three different numerical techniques for their optimization. Chapter 3, instead, focuses on the detection of mesoscale structures and, in particular, on the formalism based upon *surprise*: as the latter allows any partition of nodes to be assigned a p-value, detecting a specific, mesoscale structural organization can be understood as

the problem of finding the corresponding, most significant partition - i.e. an optimization problem whose score function is, precisely, surprise. Finally, chapter 4 deals with the application of a couple of ERGs and of the surprise-based formalism to cryptocurrencies (specifically, Bitcoin).

Chapter 1

Introduction

The first two decades of the 21st century have been characterized by a prominent rise of availability of digital data: within such a context, claims like 'Data is the new oil' [1, 2], 'Data are the most valuable resource' [3] *et similia* have become a sort of mantra - by itself, however, quite useless. In fact, (big) data (just) contribute to clarify the empirical picture concerning a (big) system, in turn calling for (big) models able to interpret/explain it.

Within such a context, network theory has emerged as a successful framework to address problems of scientific and societal relevance [4] as the spreading of infectious diseases [5, 6, 7], the dynamics of opinions [8], the propagation of losses during financial crises (the so-called 'cascading failures') [9, 10, 11], the reaction of a system against a targeted attack [12, 13].

The aforementioned examples let two methodological needs emerge quite naturally [14]: 1) a first one, that can be described as the detection of the topological properties of a given network structure that can be deemed as statistically-significant - typically, the higher-order properties (as the assortativity and the clustering coefficient) that local features of the nodes (e.g. the degree) cannot explain; 2) a second one, that can be described as the inference of the relevant details of a networked configuration in case only partial information about it is available.

Both goals can be achieved by constructing a framework for defining *benchmarks*, i.e. synthetic configurations retaining only some of the properties of the original system - the so-called *constraints* - but, otherwise, being maximally random. To tackle the aforementioned problems, two different kinds of approaches have been proposed so far, i.e. the *microcanonical* and the *canonical* ones. Microcanonical approaches [15, 16, 17, 18, 19, 20, 21] enforce the constraints in a 'hard' fashion, by (numerically) generating many randomized variants of the empirical configuration on each of which the constrained properties are identical to the empirical ones. On the other hand, canonical approaches [22, 23, 24, 25, 26] enforce constraints in a 'soft' fashion, by creating a set of configurations over which the constrained properties are identical to the empirical ones only *on average*. Softening the requirement of matching the constraints has a clear advantage: it allows the mathematical expression for the probability of observing a generic configuration, $P(\mathbf{G})$, to be obtained analytically, as a function of the enforced constraints.

The oldest, random network model is the one defined by Erdős and Rényi, also known as Random Graph Model (RGM). Introduced in 1959 [27] and closely related to Gilbert's model [28], the Random Graph Model, denoted as $G(N, L)$, represents a suitable benchmark to analyze networks with N nodes and L edges. While, in its original formulation, the RGM is microcanonical - i.e. all networks with N nodes and L edges are assigned the same probability - Gilbert's model is, instead, canonical and is entirely defined by the probability of observing an edge, p, which is the same for each pair of nodes. One of the (undesirable) consequences of Gilbert's model is that of predicting degrees which are normally distributed around the expected value $\langle k \rangle = p(N-1)$ as the total number of nodes, N, becomes large enough. In fact, this result represents the major drawback of the RGM preventing it from reproducing one of the main features characterizing real-world graphs, i.e. the appearance of *scale-free*[1] degree distributions.

[1] Mathematically speaking, scale-free degree distributions are power-laws whose exponent lies between 2 and 3: since the m-th moment is finite if $m < \gamma - 1$, the second moment diverges if $2 < \gamma < 3$, i.e. precisely for the range of values characterizing many real-world networks.

The RGM can control only for the number of edges in the network but not for local properties such as the number of connections any node establishes with other nodes. From an historical point of view, a step forward in this direction was made by Holland and Leinhardt, whose model [29] extends the Erdős-Rényi one, by constraining the in-degrees, the out-degrees and the number of reciprocal links (in fact, it depends on $2N+1$ parameters and, as the RGM, is characterized by conditionally independent edges): models of the kind are known as *Exponential Random Graphs* (ERGs), as Frank and Strauss named them[2] [30].

Although we have mentioned just few examples, the ones we have cited are sufficient to highlight a general trend in statistical modeling: the more information is considered, the more parameters are required to encapsulate it. In general, these parameters cannot be computed analytically, whence the need to rest upon numerical methods. In a scenario like this one, defining a proper null model for a network with thousands of nodes can become computationally expensive - and things get even worse when larger graphs are considered: the evidence that the most interesting real-world networks (i.e. Facebook, Twitter and Bitcoin) are all characterized by hundred of thousands/millions of nodes leads to the conclusion that recipes for the numerical resolution of the aforementioned models that are both fast and scalable are needed. The second chapter of the thesis is devoted to illustrate our contribution to fill this gap, by introducing iterative methods for the resolution of ERGs[3].

Besides, we will deal with one of the most common applications of null models, i.e. the detection of statistically-significant mesoscale structures such as modular (i.e. *communities*) and bimodular (i.e. *core-periphery* and *bipartite*) ones. To this aim, many algorithms have been developed, using methods and tools borrowed from different disciplines [32] [33] [34] [35] [36] [37] [38]; despite this, a comprehensive, theoretically-sound framework for detecting *all kinds* of mesoscale structures is still missing

[2]For a detailed discussion about ERGs, the interested reader is redirected to the second chapter of the present thesis.

[3]A Python library to solve ERGs for undirected, directed, binary and weighted, monopartite networks and for undirected, binary, bipartite networks has been released together with the paper [31].

[39, 40]. The third chapter of the thesis is devoted to illustrate our contribution to fill this gap, by generalizing the *surprise* function [41, 33, 38] to handle all the aforementioned cases[4].

Lastly, the fourth chapter is devoted to the application of the methods described in chapters 2 and 3 to the Bitcoin Lightning Network (BLN). The choice of considering such a particular system has been driven by two considerations: 1) the BLN is characterized by a growing number of users, hence can be employed to test the scalability of the algorithms for solving ERGs introduced in chapter 2; 2) the BLN is weighted, hence can be employed to test all versions of surprise (in particular, the weighted ones introduced in chapter 3) to, e.g. individuate the presence of a core-periphery structure, check if the presence of weights impacts on the latter and how, etc.

[4]A Python library to implement all variants of surprise discussed in the present thesis has been released together with the paper [42].

Chapter 2

Fast and scalable likelihood maximization for ERGs with local constraints

This chapter is devoted to present the results of the paper [31], published on Scientific Reports and dealing with the definition of iterative, fast and scalable methods to solve ERGs with local constraints. After recalling the theoretical foundations of such a framework (i.e. entropy and likelihood maximization), we approach the problem from a purely numerical perspective and consider three algorithms for the estimation of the parameters that define locally-constrained ERGs, in both a binary and a weighted fashion. Finally, we compare and discuss the performances of these algorithms on different real-world networks.

2.1 Introduction

As we were mentioning in the introductory chapter, the problem of generating benchmarks for networks constrained to reproduce specific properties but, otherwise, maximally random, can be tackled by adopting two different approaches: either a *microcanonical* or a *canonical* one.

The microcanonical approach assigns the same probability to all configurations on which the constrained properties are preserved exactly and zero probability to all the other, possible configurations (i.e. the ones that do not satisfy such constraints): in other words, the admissible, equiprobable configurations are those on which constraints are satisfied in a 'hard' fashion.

The canonical approach, instead, relaxes the aforementioned restriction and prescribes to preserve the network properties of interest on average: this implies that every, possible configuration - even those on which the constrained properties differ from their empirical counterparts - is characterized by a probability of being observed, $P(\mathbf{G})$, that is different from zero [4, 22].

The Exponential Random Graphs formalism [43], whose popularity has steadily risen over the years, allows one to implement the canonical approach to study networks. It dates back to Gibbs' (re)formulation of statistical mechanics and is based upon the variational principle named *maximum entropy*, stating that the probability distribution that is maximally non-committal with respect to the missing information is the one maximizing *Shannon entropy* [44]. This allows self-consistent inference to be made by assuming maximal ignorance about the unknown degrees of freedom of the system.

2.2 The Maximum Entropy Principle

Defining the null model to be employed as a benchmark boils down to finding the solution, $P(\mathbf{G})$, of the following maximization problem

$$\arg\max_{P} \; S[P] \tag{2.1a}$$

$$\text{s.t.} \; \sum_{\mathbf{G}} P(\mathbf{G})C_i(\mathbf{G}) = \langle C_i \rangle, \quad i = 0 \dots M \tag{2.1b}$$

where Shannon entropy reads

$$S[P] = -\sum_{\mathbf{G}} P(\mathbf{G}) \ln P(\mathbf{G}) \tag{2.2}$$

and $\vec{C}(\mathbf{G})$ is the vector of constraints representing the information defining the benchmark itself (notice that $C_0 = \langle C_0 \rangle = 1$ sums up the normalization condition). Finding the solution of the problem 2.1 is equivalent at maximizing the *Lagrangian function*

$$\mathcal{L}(P, \vec{\theta}) \equiv S[P] + \sum_{i=0}^{M} \theta_i \left[-\sum_{\mathbf{G}} P(\mathbf{G}) C_i(\mathbf{G}) + \langle C_i \rangle \right] \tag{2.3}$$

with respect to $P(\mathbf{G})$. As a result, one obtains

$$P(\mathbf{G}|\vec{\theta}) = \frac{e^{-\mathcal{H}(\mathbf{G},\vec{\theta})}}{Z(\vec{\theta})} \tag{2.4}$$

with $\mathcal{H}(\mathbf{G}, \vec{\theta}) = \vec{\theta} \cdot \vec{C}(\mathbf{G}) = \sum_{i=1}^{M} \theta_i C_i(\mathbf{G})$ representing the *Hamiltonian*, i.e. the function summing up the proper, imposed constraints and $Z(\vec{\theta}) = \sum_{\mathbf{G}} P(\mathbf{G}) = \sum_{\mathbf{G}} e^{-\mathcal{H}(\mathbf{G},\vec{\theta})}$ representing the *partition function*, ensuring that $P(\mathbf{G})$ is properly normalized. The role played by constraints is crucial and strongly dependent on the particular use-case: when the analysis aims at assessing the significance of specific quantities, the constraints represent the information that we are filtering out; on the other hand, if our objective is that of reconstructing the (inaccessible) details of a given configuration, the constraints represent the only, available information.

2.3 The Maximum Likelihood Principle

The formalism above is perfectly general; however, it can be instantiated to study an empirical network configuration, say $\mathbf{G}^*$. In this case, the Lagrange multipliers 'acting' as unknown parameters in eq. (2.4) can be numerically estimated by maximizing the log-likelihood function (associated with) $P(\mathbf{G}|\vec{\theta})$ [22, 45]. The latter is defined as

$$\mathscr{L}(\vec{\theta}) \equiv \ln P(\mathbf{G}^*|\vec{\theta}) = -\mathcal{H}(\mathbf{G}^*, \vec{\theta}) - \ln Z(\vec{\theta}) \tag{2.5}$$

and must be maximized with respect to the vector $\vec{\theta}$. Whenever the probability distribution is exponential (as the one deriving from Shannon entropy maximization), the likelihood maximization problem

$$\arg\max_{\vec{\theta}} \ \mathscr{L}(\vec{\theta}) \tag{2.6}$$

is characterized by first-order, necessary conditions for optimality reading

$$\begin{aligned} \frac{\partial \mathscr{L}(\vec{\theta})}{\partial \theta_i} &= -C_i(\mathbf{G}^*) - \frac{\partial \ln Z(\vec{\theta})}{\partial \theta_i} \\ &= -C_i(\mathbf{G}^*) + \sum_{\mathbf{G}} C_i(\mathbf{G}) P(\mathbf{G}) \\ &= -C_i(\mathbf{G}^*) + \langle C_i \rangle = 0, \quad i = 1 \dots M \end{aligned} \tag{2.7}$$

and leading to the system of equations

$$\nabla \mathscr{L}(\vec{\theta}) = \frac{\partial \mathscr{L}(\vec{\theta})}{\partial \vec{\theta}} = \vec{0} \Longrightarrow \vec{C}(\mathbf{G}^*) = \langle \vec{C} \rangle \tag{2.8}$$

to be solved. These conditions, however, are sufficient to characterize a maximum only if $\mathscr{L}(\vec{\theta})$ is concave. This is indeed the case, as we prove by noticing that

$$H_{ij} = \frac{\partial^2 \mathscr{L}(\vec{\theta})}{\partial \theta_i \partial \theta_j} = -\frac{\partial^2 \ln Z(\vec{\theta})}{\partial \theta_i \partial \theta_j} = \frac{\partial \langle C_j \rangle}{\partial \theta_i} = -\text{Cov}[C_i, C_j], \quad i, j = 1 \dots M \tag{2.9}$$

i.e. that the Hessian matrix, $\mathbf{H}$, of our likelihood function is 'minus' the covariance matrix of the constraints, hence negative semidefinite by definition[1] [43]. A graphical representation of how the two principles work is shown in fig. 1.

[1]The third passage is an example of the well-known *fluctuation-response relation*.

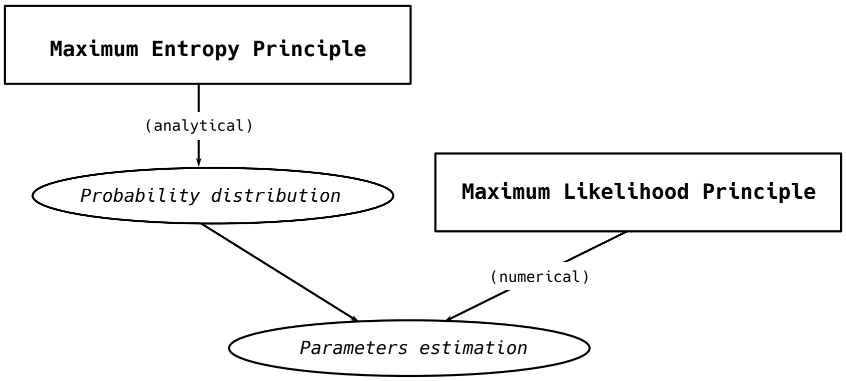


Figure 1: Graphical visualization of how the MEP and the MLP work: while the MEP allows the functional form of a probability distribution to be determined analytically, the MLP provides the recipe to numerically determine the parameters defining it.

2.4 Combining the MEP and the MLP

The Maximum Entropy Principle (MEP) and the Maximum Likelihood Principle (MLP) encode two different prescriptions aiming, respectively, at determining the functional form of a probability distribution and its numerical value.

In optimization theory, the problem 2.1 is known as *primal problem*: upon noticing that Shannon entropy is concave, while the imposed constraints are linear in $P(\mathbf{G})$, one concludes that the primal problem is convex (it is easy to see this, by rewriting it as a minimization problem for $-S[P]$).

As convexity implies *strong duality*, we can, equivalently, consider an alternative version of the problem to optimize, know as *dual problem*. In order to define it, let us consider the Lagrangian function

$$\mathcal{L}(P,\vec{\theta}) \equiv S[P] + \sum_{i=1}^{M} \theta_i \left[-\sum_{\mathbf{G}} P(\mathbf{G}) C_i(\mathbf{G}) + C_i(\mathbf{G}^*) \right] \tag{2.10}$$

where, now, the generic expectation of the i-th constraint, $\langle C_i \rangle$, has been replaced by the corresponding empirical value, $C_i(\mathbf{G}^*)$. As the dual function is given by

$$P(\mathbf{G}^*|\vec{\theta}) \equiv \arg\max_P \mathcal{L}(P, \vec{\theta}), \tag{2.11}$$

the dual problem reads

$$\arg\max_{\vec{\theta}} \arg\min_P -\mathcal{L}(P(\vec{\theta}), \vec{\theta}) \tag{2.12}$$

which is a convex problem by construction; this is readily seen by substituting eq. (2.4) into eq. (2.10), an operation that leads to the expression

$$-\mathcal{L}(P(\vec{\theta}), \vec{\theta}) = -\vec{\theta} \cdot \vec{C}(\mathbf{G}^*) - \ln Z(\vec{\theta}) = \mathscr{L}(\vec{\theta}), \tag{2.13}$$

i.e. the likelihood function introduced in eq. (2.5). In other words, eq. (2.12) combines the MEP and the MLP into a unique optimization step whose score function becomes the Lagrangian function defined in eq. (2.10).

2.5 Optimization algorithms for non-linear problems

In general, the optimization problem defined in eq. (2.12) cannot be solved analytically, whence the need to resort to numerical methods[2]. The problem

$$\arg\max_{\vec{\theta}} \mathscr{L}(\vec{\theta}) \tag{2.14}$$

[2]For an exhaustive review on numerical methods for optimization we refer the interested reader to [46, 47]: in the following, we present only the concepts that are of some relevance for us.

is a Non-linear Programming Problem (NLP) that can be solved numerically by adopting a Sequential Quadratic Programming (SQP) approach. Starting from an initial guess $\vec{\theta}^{(0)}$, SQP iteratively updates the vector of Lagrange multipliers

$$\vec{\theta}_i^{(n+1)} = \vec{\theta}_i^{(n)} + \alpha\Delta\vec{\theta}_i^{(n)}, \quad i = 1 \dots M \tag{2.15}$$

according to the rule

$$\Delta\vec{\theta}_i^{(n)} = \underset{\Delta\vec{\theta}_i}{\arg\max} \left[\nabla_{\vec{\theta}_i}\mathscr{L}(\vec{\theta})\Delta\vec{\theta}_i + \sum_{j,k}\frac{1}{2}\Delta\vec{\theta}_j H_{jk}^{(n)}\Delta\vec{\theta}_k\right], \quad \forall\, i \tag{2.16}$$

which leads to the set of equations

$$\nabla_i\mathscr{L}(\vec{\theta}) + \sum_j H_{ij}^{(n)}\Delta\vec{\theta} = 0, \quad i = 1 \dots M \tag{2.17}$$

that can be compactly rewritten as

$$\Delta\vec{\theta}^{(n)} = -{\mathbf{H}^{(n)}}^{-1}\nabla\mathscr{L}(\vec{\theta}); \tag{2.18}$$

the stepsize $\alpha \in (0,1]$ is selected to ensure that $\mathscr{L}(\vec{\theta}^{(n+1)}) > \mathscr{L}(\vec{\theta}^{(n)})$ via a back-tracking, line-search procedure: starting from $\alpha = 1$, if the Armijo condition

$$\mathscr{L}(\vec{\theta}^{(n)} + \alpha\Delta\vec{\theta}^{(n)}) < \mathscr{L}(\vec{\theta}^{(n)}) + \gamma\alpha\nabla\mathscr{L}(\vec{\theta})^\top\Delta\vec{\theta}, \tag{2.19}$$

is violated, we set $\alpha \leftarrow \beta\alpha$ - where $\gamma \in (0, 0.5]$ and $\beta \in (0,1)$ are the parameters of the algorithm.

The term $\mathbf{H}^{(n)}$, instead, can be selected according to a variety of methods: in the present contribution we focus on the following three ones.

2.5.1 Newton's method

One speaks of Newton's method in case $\mathbf{H}^{(n)}$ is chosen to be

$$\mathbf{H}^{(n)} = \nabla^2 \mathscr{L}(\vec{\theta}^{(n)}) + \Delta\mathbf{H}^{(n)} \tag{2.20}$$

where $\nabla^2 \mathscr{L}(\vec{\theta})$ is the Hessian matrix of the likelihood function and the term $\Delta\mathbf{H}^{(n)}$ is typically selected as small as possible in order to avoid slowing convergence - however, still ensuring that $\mathbf{H}^{(n)}$ is negative definite (i.e. $\nabla^2 \mathscr{L}(\vec{\theta}^{(n)}) + \Delta\mathbf{H}^{(n)} \prec 0$). This choice of $\mathbf{H}^{(n)}$ is also referred to as 'exact Hessian'.

2.5.2 Quasi-Newton methods

Any Hessian approximation which is negative definite (i.e. satisfying $\mathbf{H}^{(n)} \prec 0$) yields an ascent direction and guarantees convergence. Although one may choose to consider the simplest prescription $\mathbf{H}^{(n)} = -\mathbf{I}$, which yields the 'steepest ascent' algorithm, here we have opted for the following recipe

$$H_{ii}^{(n)} = \nabla_{ii}^2 \mathscr{L}(\vec{\theta}^{(n)}) + \Delta H_{ii}^{(n)} < 0, \quad \forall\, i \tag{2.21}$$

and

$$H_{ij}^{(n)} = 0, \quad \forall\, i \neq j \tag{2.22}$$

i.e. the purely diagonal version of Newton's method.

2.5.3 Fixed-point iteration on modified KKT conditions

In addition to the (classes of) algorithms described above, we will also consider an iterative recipe which is constructed as a fixed-point iteration on a modified version of the Karush–Kuhn–Tucker (KKT) conditions, i.e.

$$\mathbf{F}(\vec{\theta}) = \vec{0} \tag{2.23}$$

or, analogously,

$$\vec{\theta} = \mathbf{G}(\vec{\theta}); \tag{2.24}$$

the iterate can, then, be made explicit by rewriting the latter as

$$\vec{\theta}^{(n)} = \mathbf{G}(\vec{\theta}^{(n-1)}). \tag{2.25}$$

The condition above, yielding a non-standard SQP method (in fact, $\mathbf{H}^{(n)}$ is typically not symmetric, for our models) will be made explicit, for each network model, in the corresponding subsection.

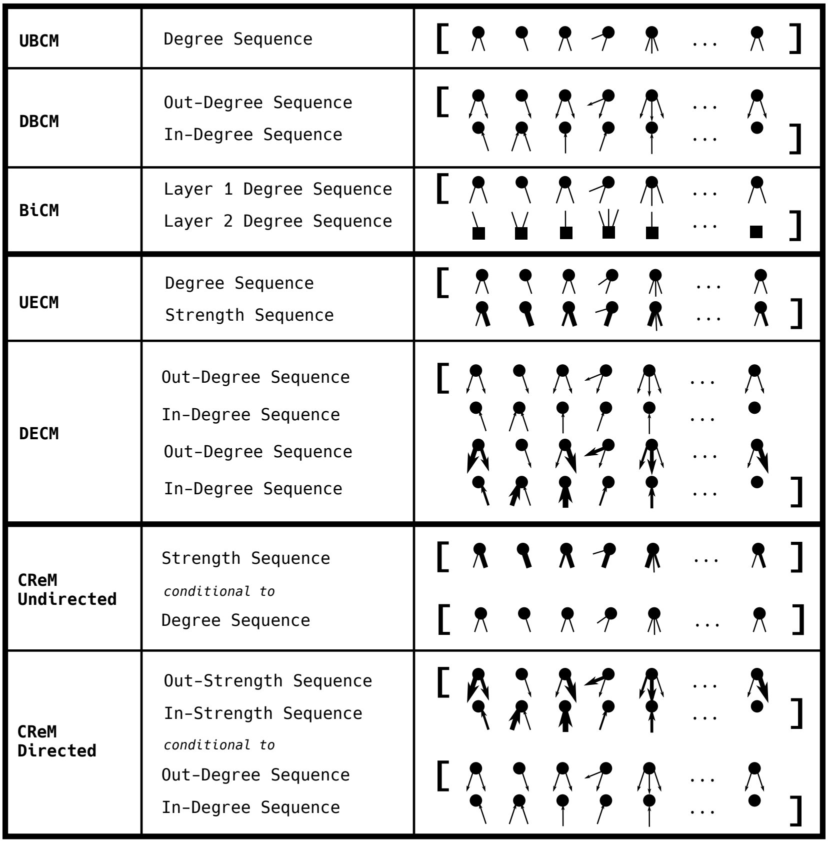
Since the Hessian approximation $\mathbf{H}^{(n)}$ is negative definite, the direction $\Delta\vec{\theta}$ is an ascending one: as such, it is guaranteed to yield an improvement of the objective function for a step size α that is sufficiently small; the back-tracking line-search guarantees the finding of a step size α that yields such an improvement, while making sufficient progresses towards the solution. As discussed in [46], Newton's method has local, quadratic convergence, while the quasi-Newton method and the fixed-point iteration algorithm have local, linear convergence.

2.6 Applications

Let us now apply the algorithms described in the previous section to a number of specific cases of interest. The constraints defining each model are illustrated in fig. 2.

2.6.1 Binary undirected graphs with given degree sequence (UBCM)

Let us start by considering binary, undirected networks (BUNs). The simplest, non-trivial set of constraints is represented by the degrees of nodes: the degree of node i, i.e. $k_i(\mathbf{A}) = \sum_{j(\neq i)=1}^{N} a_{ij}$, counts the number of its neighbours and coincides with the total number of 1s along the i-th row (or, equivalently, along the i-th column) of the adjacency matrix $\mathbf{A} \equiv$



UBCM	Degree Sequence
DBCM	Out-Degree Sequence In-Degree Sequence
BiCM	Layer 1 Degree Sequence Layer 2 Degree Sequence
UECM	Degree Sequence Strength Sequence
DECM	Out-Degree Sequence In-Degree Sequence Out-Degree Sequence In-Degree Sequence
CReM Undirected	Strength Sequence *conditional to* Degree Sequence
CReM Directed	Out-Strength Sequence In-Strength Sequence *conditional to* Out-Degree Sequence In-Degree Sequence

Figure 2: Graphical visualization of the constraints defining (some of) the ERGs considered in this chapter. Notice that while the enhanced models (i.e. UECM and DECM) constrain binary and weighted quantities in a joint fashion, the conditional models (i.e. the CReM ones) allow for a 'separate' specification of them.

$\{a_{ij}\}_{i,j=1}^{N}$. The benchmark defined by this set of constraints is known as *Undirected Binary Configuration Model* (UBCM) and its Hamiltonian reads

$$\mathcal{H}_{\text{UBCM}}(\mathbf{A},\vec{\theta}) = \sum_{i=1}^{N} \theta_i k_i(\mathbf{A}); \tag{2.26}$$

entropy maximization [4, 22] leads to the factorized graph probability

$$P_{\mathrm{UBCM}}(\mathbf{A}|\vec{\theta}) = \prod_{i=1}^{N} \prod_{\substack{j=1 \\ (j<i)}}^{N} p_{ij}^{a_{ij}} (1 - p_{ij})^{1-a_{ij}} \tag{2.27}$$

where $p_{ij} = p_{ij}^{\mathrm{UBCM}} \equiv \frac{e^{-\theta_i - \theta_j}}{1+e^{-\theta_i-\theta_j}}$. In this case, the canonical ensemble of BUNs is the set of networks with the same number of nodes, N, of the observed graph and a number of (undirected) links varying from zero to the maximum value $\binom{N}{2}$.

The argument of the problem 2.14 for the specific network $\mathbf{A}^*$ becomes

$$\mathscr{L}_{\mathrm{UBCM}}(\vec{\theta}) = -\sum_{i=1}^{N} \theta_i k_i(\mathbf{A}^*) - \sum_{i=1}^{N} \sum_{\substack{j=1 \\ (j<i)}}^{N} \ln\left[1 + e^{-\theta_i - \theta_j}\right] \tag{2.28}$$

whose first-order, optimality conditions read

$$\begin{aligned}
\nabla_{\theta_i} \mathscr{L}_{\mathrm{UBCM}} &= -k_i(\mathbf{A}^*) + \sum_{\substack{j=1 \\ (j\neq i)}}^{N} \frac{e^{-\theta_i-\theta_j}}{1+e^{-\theta_i-\theta_j}} \\
&= -k_i(\mathbf{A}^*) + \sum_{\substack{j=1 \\ (j\neq i)}}^{N} p_{ij}^{\mathrm{UBCM}} \\
&= -k_i(\mathbf{A}^*) + \langle k_i \rangle = 0, \quad i = 1 \dots N.
\end{aligned} \tag{2.29}$$

Resolution of the UBCM

Newton's and the quasi-Newton method can be easily implemented via the recipe defined in eq. (2.18) (see Appendix A.1 for the definition of the UBCM Hessian).

The explicit definition of the fixed-point recipe, instead, requires a preliminary observation, i.e. that the system of equations embodying the UBCM first-order, optimality conditions can be re-written as follows

$$e^{-\theta_i} = \frac{k_i(\mathbf{A}^*)}{\sum_{\substack{j=1\\(j\neq i)}}^{N}\left(\frac{e^{-\theta_j}}{1+e^{-\theta_i-\theta_j}}\right)}, \quad i = 1 \dots N \tag{2.30}$$

i.e. as a set of consistency equations. The observation that the term $e^{-\theta_i}$ appears on both sides of the equation corresponding to the i-th constraint suggests an iterative recipe to solve such a system, i.e.

$$\theta_i^{(n)} = -\ln\left[\frac{k_i(\mathbf{A}^*)}{\sum_{\substack{j=1\\(j\neq i)}}^{N}\left(\frac{e^{-\theta_j^{(n-1)}}}{1+e^{-\theta_i^{(n-1)}-\theta_j^{(n-1)}}}\right)}\right], \quad i = 1 \dots N \tag{2.31}$$

originally proposed in [48] and further refined in [49]. The identification $p_{ij}^{\text{UBCM}} \equiv \frac{e^{-\theta_i^{(\infty)}-\theta_j^{(\infty)}}}{1+e^{-\theta_i^{(\infty)}-\theta_j^{(\infty)}}}$, $\forall\, i < j$ allows the probability coefficients defining the UBCM to be numerically determined.

As any other iterative recipe, the one proposed above needs to be initialized as well. To this aim, we have tested three different sets of initial values: the first one is defined by the position $\theta_i^{(0)} = -\ln\left[\frac{k_i(\mathbf{A}^*)}{\sqrt{2L}}\right]$, $\forall\, i$ - usually, a good approximation of the solution of the system of equations in (2.29), in the 'sparse case' (i.e. whenever $p_{ij}^{\text{UBCM}} \simeq e^{-\theta_i-\theta_j}$ [50]); the second one is a variant of the position above, reading $\theta_i^{(0)} = -\ln\left[\frac{k_i(\mathbf{A}^*)}{\sqrt{N}}\right]$, $\forall\, i$; the third one, instead, prescribes to randomly draw the value of each parameter from a uniform distribution with support on the unit interval, i.e. $\theta_i^{(0)} \sim \text{U}(0,1)$, $\forall\, i$.

Reducing the dimensionality of the problem

The problem defining the UBCM can be further simplified by noticing that nodes with the same degree, say k, can be assigned the same value of the multiplier θ [45] - a result resting upon the observation that any value $k_i(\mathbf{A}^*)$ must match the sum of monotonic, increasing functions. This translates into the possibility of rewriting $\mathscr{L}_{\text{UBCM}}(\vec{\theta})$ in a 'reduced' fashion, as

$$
\begin{aligned}
\mathscr{L}^{\text{reduced}}_{\text{UBCM}}(\vec{\theta}) = & -\sum_{k} f(k)\theta_k k(\mathbf{A}^*) \\
& -\sum_{k} \sum_{\substack{k' \\ (k' \leq k)}} f(k)[f(k') - \delta_{kk'}] \ln \left[1 + e^{-\theta_k - \theta_{k'}}\right]
\end{aligned}
\tag{2.32}
$$

where the sums run over the *distinct* values of the degrees and $f(k)$ counts the number of nodes whose degree is k. Rewriting the problem with respect to the set $\{\theta_k\}_k$ leads one to recover simplified versions of the three algorithms considered here: Newton's and the quasi-Newton methods can, now, be solved via a 'reduced' version of eq. (2.18) - since both the dimension of the gradient and the order of the Hessian matrix of the likelihood function are, now, less than N - while the iterative recipe defined in (2.30) can be rewritten in terms of the 'non-degenerate' degrees, as

$$
\theta_k^{(n)} = -\ln \left[\frac{k(\mathbf{A}^*)}{\sum_{k'} [f(k') - \delta_{kk'}] \left(\frac{e^{-\theta_{k'}^{(n-1)}}}{1+e^{-\theta_k^{(n-1)} - \theta_{k'}^{(n-1)}}} \right)} \right], \quad \forall\, k \tag{2.33}
$$

where, at the denominator, the self-contribution (i.e. the probability that a node links to itself) has been explicitly excluded.

Performance testing

The accuracy of each algorithm in reproducing the constraints defining the UBCM has been quantified via the *maximum absolute error* metrics, defined, in a perfectly general fashion, as $\max_i\{|C_i^* - \langle C_i \rangle|\}_{i=1}^N$ (where C_i^* is the empirical value of the i-th constraint, C_i). Naturally, in the UBCM case, $C_i = k_i$, $\forall\, i$ and the aforementioned error score becomes

$$
\text{MADE} = \max_i\{|k_i^* - \langle k_i \rangle|\}_{i=1}^N \tag{2.34}
$$

(the acronym standing for Maximum Absolute Degree Error). Equivalently, it is the infinite norm of the difference between the vector of the empirical values of the constraints and that of their expected values.

For each algorithm, we have also considered three different stopping criteria: the first one puts a condition on the Euclidean norm of the gradient of the likelihood function, i.e.

$$||\nabla\mathscr{L}(\vec{\theta})||_2 = \sqrt{\sum_{i=1}^{N}\left(\nabla_i\mathscr{L}(\vec{\theta})\right)^2} \leq 10^{-8}; \tag{2.35}$$

the second one puts a condition on the Euclidean norm of the vector of differences between the values of the parameters at subsequent iterations, i.e.

$$||\Delta\vec{\theta}||_2 = \sqrt{\sum_{i=1}^{N}(\Delta\theta_i)^2} \leq 10^{-8}; \tag{2.36}$$

the third one concerns the maximum number of iterations: after 1.000 steps, any of the three algorithms stops.

Results

The performance of the three algorithms, considered in the present paper, to solve the reduced version of eqs. (2.29), has been tested on a bunch of real-world networks. The latter span a wide variety of systems, including natural, financial and technological ones. In particular, we have considered the synaptic network of the worm *C. Elegans* [51], the network of the largest US airports [52], the protein-protein interaction network of the bacterium *H. Pylori* [53], Internet at the level of Autonomous Systems [54] and eight, daily snapshots of the so-called Bitcoin Lightning Network [55] (chosen throughout its entire history).

The results about the performance of our three algorithms are reported in Table 1. Overall, all recipes perform very satisfactorily, being accurate, fast and scalable; moreover, all algorithms stop either because the condition on the norm of the likelihood is satisfied or because the condition on the norm of the vector of parameters is satisfied.

For what concerns accuracy, the largest maximum error per method spans an interval (across all configurations) that amounts at $10^{-10} \lesssim \text{MADE}^{\text{reduced}}_{\text{Newton}} \lesssim 10^{-6}$, $10^{-6} \leq \text{MADE}^{\text{reduced}}_{\text{Quasi-Newton}} \leq 10^{-5}$ and $10^{-8} \lesssim \text{MADE}^{\text{reduced}}_{\text{fixed-point}} \lesssim 10^{-6}$. By looking at each specific network, it is evident that the two most accurate methods are systematically Newton's and the fixed-point ones.

For what concerns speed, the amount of time (measured in seconds) required by each method to achieve convergence spans an interval (across all configurations) that is $0.005 \leq T^{\text{reduced}}_{\text{Newton}} \leq 0.01$, $0.014 \leq T^{\text{reduced}}_{\text{Quasi-Newton}} \leq 0.15$ and $0.002 \leq T^{\text{reduced}}_{\text{fixed-point}} \leq 0.015$. The fastest method is the fixed-point one, although Newton's method approximately requires the same amount of time, when compared to it on each specific configuration. Differences in the speed of convergence of any method, caused by the choice of a particular set of initial conditions, are indeed observable: the prescription reading $\theta_i^{(0)} = -\ln\left[\frac{k_i(\mathbf{A}^*)}{\sqrt{N}}\right], \forall\, i$ outperforms the other ones.

Let us now comment on the scalability of our algorithms. What we learn from our exercise is that scalability is not related to the network size in a simple way: the factors (seemingly) playing a major role are the ones affecting the reducibility of the original system of equations, i.e. the ones 'deciding' the number of different equations that actually need to be solved. While reducibility can be easily quantified *a posteriori*, e.g. by calculating the *coefficient of reduction*, c_r, defined as the ratio between the number of equations that survive to reduction and the number of equations defining the original problem (hence, the smaller the better), providing an exhaustive list of the aforementioned factors *a priori* is much more difficult.

In the case of the UBCM, c_r is defined as the number of different degrees divided by the total number of nodes; one may, thus, argue that reducibility is affected by the heterogeneity of the degree distribution; upon considering that the latter can be quantified by computing the *coefficient of variation* (defined as $c_v = s/m$, where s and m are, respectively, the standard deviation and the mean of the degree distribution of the network at hand), one may derive a simple rule of thumb: a larger coefficient of variation (pointing out a larger heterogeneity of the degree dis-

					Newton		Quasi-Newton		Fixed-point	
	N	L	c	c_r	MADE	Time (s)	MADE	Time (s)	MADE	Time (s)
C. Elegans (nn)	265	1879	$\simeq 5 \cdot 10^{-2}$	$\simeq 1.5 \cdot 10^{-1}$	$\simeq 8.1 \cdot 10^{-8}$	$\simeq 0.005$	$\simeq 1 \cdot 10^{-6}$	$\simeq 0.03$	$\simeq 5 \cdot 10^{-8}$	$\simeq 0.004$
US airports	500	2980	$\simeq 2 \cdot 10^{-2}$	$\simeq 1.3 \cdot 10^{-1}$	$\simeq 8.9 \cdot 10^{-9}$	$\simeq 0.008$	$\simeq 1.2 \cdot 10^{-6}$	$\simeq 0.04$	$\simeq 2.5 \cdot 10^{-7}$	$\simeq 0.005$
H. Pylori (pp)	732	1465	$\simeq 5 \cdot 10^{-3}$	$\simeq 4.5 \cdot 10^{-2}$	$\simeq 1.3 \cdot 10^{-8}$	$\simeq 0.004$	$\simeq 6.7 \cdot 10^{-7}$	$\simeq 0.03$	$\simeq 7 \cdot 10^{-8}$	$\simeq 0.014$
Internet (AS)	11174	23409	$\simeq 4 \cdot 10^{-4}$	$\simeq 1 \cdot 10^{-2}$	$\simeq 4.1 \cdot 10^{-7}$	$\simeq 0.03$	$\simeq 5.1 \cdot 10^{-6}$	$\simeq 0.10$	$\simeq 2 \cdot 10^{-6}$	$\simeq 0.005$
BLN 24-01-18	94	152	$\simeq 3 \cdot 10^{-3}$	$\simeq 1.5 \cdot 10^{-1}$	$\simeq 1 \cdot 10^{-8}$	$\simeq 0.005$	$\simeq 1.7 \cdot 10^{-6}$	$\simeq 0.014$	$\simeq 4 \cdot 10^{-8}$	$\simeq 0.002$
BLN 25-02-18	499	1010	$\simeq 8 \cdot 10^{-3}$	$\simeq 6.4 \cdot 10^{-2}$	$\simeq 1.6 \cdot 10^{-8}$	$\simeq 0.005$	$\simeq 9.1 \cdot 10^{-7}$	$\simeq 0.02$	$\simeq 7.3 \cdot 10^{-8}$	$\simeq 0.004$
BLN 30-03-18	1012	2952	$\simeq 5 \cdot 10^{-3}$	$\simeq 5.2 \cdot 10^{-2}$	$\simeq 3.4 \cdot 10^{-10}$	$\simeq 0.005$	$\simeq 2 \cdot 10^{-5}$	$\simeq 0.03$	$\simeq 1.4 \cdot 10^{-7}$	$\simeq 0.005$
BLN 13-07-18	1999	8999	$\simeq 4 \cdot 10^{-3}$	$\simeq 3.8 \cdot 10^{-2}$	$\simeq 6.7 \cdot 10^{-10}$	$\simeq 0.01$	$\simeq 3 \cdot 10^{-6}$	$\simeq 0.05$	$\simeq 1.7 \cdot 10^{-7}$	$\simeq 0.008$
BLN 19-12-18	3007	17689	$\simeq 4 \cdot 10^{-3}$	$\simeq 4.5 \cdot 10^{-2}$	$\simeq 1.7 \cdot 10^{-6}$	$\simeq 0.03$	$\simeq 1 \cdot 10^{-5}$	$\simeq 0.09$	$\simeq 2 \cdot 10^{-7}$	$\simeq 0.010$
BLN 30-01-19	3996	27429	$\simeq 3 \cdot 10^{-3}$	$\simeq 4.2 \cdot 10^{-2}$	$\simeq 7.2 \cdot 10^{-7}$	$\simeq 0.04$	$\simeq 1.3 \cdot 10^{-5}$	$\simeq 0.12$	$\simeq 2.7 \cdot 10^{-7}$	$\simeq 0.012$
BLN 01-03-19	5012	41096	$\simeq 3 \cdot 10^{-3}$	$\simeq 4.7 \cdot 10^{-2}$	$\simeq 3.9 \cdot 10^{-10}$	$\simeq 0.03$	$\simeq 1.7 \cdot 10^{-4}$	$\simeq 0.13$	$\simeq 2.6 \cdot 10^{-7}$	$\simeq 0.013$
BLN 17-07-19	6447	54476	$\simeq 3 \cdot 10^{-3}$	$\simeq 3.3 \cdot 10^{-2}$	$\simeq 4.1 \cdot 10^{-6}$	$\simeq 0.06$	$\simeq 1.2 \cdot 10^{-5}$	$\simeq 0.15$	$\simeq 3.1 \cdot 10^{-7}$	$\simeq 0.015$

Table 1: Performance of Newton's, quasi-Newton and the fixed-point algorithm to solve the reduced system of equations defining the UBCM, on a set of real-world BUNs (of which basic statistics as the total number of nodes, N, the total number of links, L, and the connectance, $c = 2L/N(N-1)$, are provided). All algorithms stop either because the condition $||\nabla \mathscr{L}(\vec{\theta})||_2 \leq 10^{-8}$ is satisfied or because the condition $||\Delta\vec{\theta}||_2 \leq 10^{-8}$ is satisfied. For what concerns accuracy, the two most accurate methods are Newton's and the fixed-point ones; for what concerns speed, the fastest method is the fixed-point one (although Newton's one approximately requires the same amount of time on each specific configuration). Only the results corresponding to the best choice of initial conditions are reported.

Algorithm 1 Sampling the UBCM ensemble

```
1: for $m = 1 \dots |E|$ do
2:     $\mathbf{A} = \mathbf{0}$;
3:     for $i = 1 \dots N$ do
4:         for $j = 1 \dots N$ and $j < i$ do
5:             if RandomUniform$[0, 1] \leq p_{ij}^{\text{UBCM}}$ then
6:                 $a_{ij} = a_{ji} = 1$;
7:             else
8:                 $a_{ij} = a_{ji} = 0$;
9:             end if
10:        end for
11:    end for
12:    Ensemble$[m] = \mathbf{A}$;
13: end for
```

tribution) leads to a larger coefficient of reduction and a larger amount of time for convergence will be required. Notice that even if the degree distribution is narrow, outliers (e.g. hubs) may still play a role, forcing the corresponding parameters to assume either very large or very small values - hence, slowing down the entire convergence process.

In this sense, scalability is the result of a (non-trivial) interplay between size and reducibility. Let us take a look at Table 1: Internet is the most reducible network of our basket, although being the largest in size, while the neural network of *C. Elegans* is one of the least reducible networks of our basket, although being the second smallest one; as a consequence, the actual number of equations defining the UBCM on *C. Elegans* is $\simeq 30$ while the actual number of equations defining the UBCM on Internet is $\simeq 100$ - whence the larger amount of time to solve the latter. Remarkably, the time required by our recipes to ensure that the largest system of equations converges to the solution ranges from thousandths to tenths of seconds.

Sampling the UBCM

As a last comment, we would like to stress that, unlike several popular approximations as the Chung-Lu one [50], the generic coefficient p_{ij}^{UBCM}

always represents a proper probability, in turn implying that eq. (2.27) also provides us with a recipe to sample the canonical ensemble of BUNs, under the UBCM. Notice that the factorization of the graph probability $P_{\text{UBCM}}(\mathbf{A}|\vec{\theta})$ greatly simplifies the entire procedure, allowing a single graph to be sampled by implementing the Bernoulli trial

$$a_{ij} = \begin{cases} 0 & 1 - p_{ij}^{\text{UBCM}} \\ 1 & p_{ij}^{\text{UBCM}} \end{cases} \tag{2.37}$$

for each (undirected) pair of nodes, in either a sequential or a parallel fashion. The sampling process, whose computational complexity amounts at $O(N^2)$, can be repeated to generate as many configurations as desired. The pseudo-code for explicitly sampling the UBCM ensemble is summed up by Algorithm 1.

We explicitly acknowledge the existence of the algorithm proposed in [56] for sampling binary, undirected networks in the sparse case, i.e. whenever the Chung-Lu model is applicable. This amounts at requiring that $p_{ij}^{\text{CL}} = \frac{k_i k_j}{2L} < 1, \forall\, i < j$, a condition whose validity is seldom verified. In fact, it does not hold in several cases of interest: an example of paramount importance is provided by sparse networks whose degree distribution is scale-free. In such cases, $k_{max} \sim N^{\frac{1}{\gamma-1}}$: hence, the hubs establish a connection with probability $p_{ij}^{\text{CL}} \sim \frac{N^{\frac{2}{\gamma-1}}}{N - N^{\frac{1}{\gamma-1}}}$ that becomes larger than 1 when $2 < \gamma \leq 3$ and diverges for $\gamma \to 2$, thus leading to a strong violation of the requirement above.

2.6.2 Weighted undirected graphs with given strengths and degrees (UECM)

Let us now focus on null models for weighted networks, defined by constraining both binary and weighted quantities[3]; the simplest model of the kind is the one constraining the degrees and the strengths in an undi-

[3]Purely weighted models such as the *Undirected* and the *Directed Weighted Configuration Model* have not been considered since, as it has been proven elsewhere [57], they perform quite poorly when employed to reconstruct networks.

rected fashion. While $k_i(\mathbf{A}) = \sum_{j(\neq i)=1}^{N} a_{ij}$ counts the number of neighbors of node i, $s_i(\mathbf{W}) = \sum_{j(\neq i)=1}^{N} w_{ij}$ defines the weighted equivalent of the degree of node i, i.e. its strength - for consistency, the binary adjacency matrix can be defined via the Heaviside step function, $\Theta[.]$, i.e. as $\mathbf{A} \equiv \Theta[\mathbf{W}]$ a position indicating that $a_{ij} = 1$ if $w_{ij} > 0$, $\forall\, i < j$ and zero otherwise. This particular model is known as *Undirected Enhanced Configuration Model* (UECM) [58, 57, 59] and its Hamiltonian reads

$$\mathcal{H}_{\text{UECM}}(\mathbf{W}, \vec{\alpha}, \vec{\beta}) = \sum_{i=1}^{N}[\alpha_i k_i(\mathbf{A}) + \beta_i s_i(\mathbf{W})]; \tag{2.38}$$

it induces a probability distribution which is halfway between a Bernoulli and a geometric one [58], i.e.

$$Q_{\text{UECM}}(\mathbf{W}|\vec{\alpha}, \vec{\beta}) = \prod_{i=1}^{N} \prod_{\substack{j=1 \\ (j<i)}}^{N} q_{ij}(w_{ij}) \tag{2.39}$$

with

$$q_{ij}(w) = \begin{cases} 1 - p_{ij}^{\text{UECM}}, & w = 0 \\ p_{ij}^{\text{UECM}}(e^{-\beta_i - \beta_j})^{w-1}(1 - e^{-\beta_i - \beta_j}), & w > 0 \end{cases} \tag{2.40}$$

for any two nodes i and j such that $i < j$ and

$$p_{ij}^{\text{UECM}} = \frac{e^{-\alpha_i - \alpha_j - \beta_i - \beta_j}}{1 - e^{-\beta_i - \beta_j} + e^{-\alpha_i - \alpha_j - \beta_i - \beta_j}}, \quad \forall\, i < j. \tag{2.41}$$

Notice that the functional form above is obtained upon requiring that the weights only assume (non-negative) integer values (i.e. $w_{ij} \in [0, +\infty)$, $\forall\, i < j$). The canonical ensemble is now constituted by the weighted configurations with N nodes and a number of (undirected) links ranging between zero and the maximum value $\binom{N}{2}$.

The argument of the problem 2.14 for the specific network $\mathbf{W}^*$ now becomes

$$\mathscr{L}_{\text{UECM}}(\vec{\alpha},\vec{\beta}) = -\sum_{i=1}^{N}[\alpha_i k_i(\mathbf{A}^*) + \beta_i s_i(\mathbf{W}^*)] \\ -\sum_{i=1}^{N}\sum_{\substack{j=1\\(j<i)}}^{N} \ln\left[1 + e^{-\alpha_i-\alpha_j}\left(\frac{e^{-\beta_i-\beta_j}}{1-e^{-\beta_i-\beta_j}}\right)\right] \tag{2.42}$$

whose first-order, optimality conditions read

$$\nabla_{\alpha_i}\mathscr{L}_{\text{UECM}} = -k_i(\mathbf{A}^*) + \sum_{\substack{j=1\\(j\neq i)}}^{N} p_{ij}^{\text{UECM}} = -k_i(\mathbf{A}^*) + \langle k_i\rangle = 0 \\ \nabla_{\beta_i}\mathscr{L}_{\text{UECM}} = -s_i(\mathbf{W}^*) + \sum_{\substack{j=1\\(j\neq i)}}^{N} \frac{p_{ij}^{\text{UECM}}}{1-e^{-\beta_i-\beta_j}} = -s_i(\mathbf{W}^*) + \langle s_i\rangle = 0 \tag{2.43}$$

with $i = 1 \dots N$.

Resolution of the UECM

Newton's and the quasi-Newton methods can be easily implemented via the recipe defined in eq. (2.18) (see Appendix A.1 for the definition of the UECM Hessian).

As for the purely binary models, the fixed-point recipe for solving the UECM first-order, optimality conditions transforms the following set of consistency equations

$$\alpha_i = -\ln\left[\frac{k_i(\mathbf{A}^*)}{\sum_{\substack{j=1\\(j\neq i)}}^N\left(\frac{e^{-\alpha_j-\beta_i-\beta_j}}{1-e^{-\beta_i-\beta_j}+e^{-\alpha_i-\alpha_j-\beta_i-\beta_j}}\right)}\right], \quad i=1\dots N$$
$$\beta_i = -\ln\left[\frac{s_i(\mathbf{W}^*)}{\sum_{\substack{j=1\\(j\neq i)}}^N\left(\frac{e^{-\alpha_i-\alpha_j-\beta_j}}{(1-e^{-\beta_i-\beta_j})(1-e^{-\beta_i-\beta_j}+e^{-\alpha_i-\alpha_j-\beta_i-\beta_j})}\right)}\right], \quad i=1\dots N \tag{2.44}$$

into the usual iterative fashion, by considering the parameters at the left hand side and at the right hand side, respectively at the n-th and at the $(n-1)$-th iteration. It is important to remark that a reduced version of the iterative recipe above can indeed be written, by assigning the same pair of values (α, β) to the nodes with the same pair of values (k, s): however, the larger heterogeneity of the strengths causes this event to happen more rarely than for purely binary models such as the UBCM.

As for the purely binary cases, three different sets of initial conditions have been considered, whose definition follows from the simplest conceivable generalization of the purely binary cases. In particular, the first set of values reads $\alpha_i^{(0)} = -\ln\left[\frac{k_i(\mathbf{A}^*)}{\sqrt{2L}}\right]$, $i=1\dots N$ and $\beta_i^{(0)} = -\ln\left[\frac{s_i(\mathbf{W}^*)}{\sqrt{2W}}\right]$, $i=1\dots N$; the second set is a variant of the first, reading $\alpha_i^{(0)} = -\ln\left[\frac{k_i(\mathbf{A}^*)}{\sqrt{N}}\right]$, $i=1\dots N$ and $\beta_i^{(0)} = -\ln\left[\frac{s_i(\mathbf{W}^*)}{\sqrt{N}}\right]$, $i=1\dots N$; the third recipe, instead, prescribes to randomly draw the value of each parameter from the uniform distribution defined on the unit interval, i.e. $\alpha_i^{(0)} \sim \mathrm{U}(0,1), \forall\, i$ and $\beta_i^{(0)} \sim \mathrm{U}(0,1), \forall\, i$.

Performance testing

The accuracy of each algorithm in reproducing the constraints defining the UECM has been, now, quantified via the *maximum relative error* metrics, defined, in a perfectly general fashion, as $\max_i\left\{\frac{|C_i^*-\langle C_i\rangle|}{C_i}\right\}_{i=1}^N$

(where C_i^* is the empirical value of the i-th constraint, C_i). In the UECM case, we can define two variants of the aforementioned error, i.e.

$$\text{MRDE} = \max_i \left\{ \frac{|k_i^* - \langle k_i \rangle|}{k_i} \right\}_{i=1}^N \tag{2.45}$$

$$\text{MRSE} = \max_i \left\{ \frac{|s_i^* - \langle s_i \rangle|}{s_i} \right\}_{i=1}^N \tag{2.46}$$

(the acronyms standing for Maximum Relative Degree Error and Maximum Relative Strength Error). The reason driving this choice lies in the evidence that, in absolute terms, strengths are affected by a larger numerical error than degrees: this, however, doesn't necessarily mean that a given algorithm performs poorly, as the magnitude of an error must be always compared with the numerical value of the quantity it refers to - whence the choice of considering relative scores.

The three different 'stop criteria' we have considered for each algorithm match the ones adopted for analysing the binary cases, consisting in a condition on the Euclidean norm of the gradient of the likelihood function, i.e. $||\nabla \mathscr{L}(\vec{\theta})||_2 \leq 10^{-8}$, and in a condition on the Euclidean norm of the vector of differences between the values of the parameters at subsequent iterations, i.e. $||\Delta\vec{\theta}||_2 \leq 10^{-8}$; the third condition concerns the maximum number of iterations: after 10.000 steps, any of the three algorithms stops.

Results

The performance of the three algorithms to solve the system of equations defining the UECM has been tested on a bunch of real-world networks: in particular, we have considered the WTW during the decade 1990-2000[4] [60].

The results are reported in Table 2. Overall, two out of three algorithms (i.e. Newton's and the quasi-Newton methods) perform very sat-

[4]Since the weights defining the configurations of the WTW are real numbers, we have rounded them to the nearest integer value, before running the UECM.

isfactorily, being accurate, fast and scalable; the third one (i.e. the fixed-point recipe), instead, performs very poorly. Moreover, while Newton's method stops because the condition on the norm of the likelihood is satisfied, both the quasi-Newton and the fixed-point algorithms are always found to satisfy the limit condition on the number of steps (i.e. they run for 10.000 steps and, then, stop).

For what concerns accuracy, the largest maximum error made by Newton's method (across all configurations) amounts at $10^{-10} \lesssim \text{MRDE}_{\text{Newton}} \lesssim 10^{-5}$ and $\text{MRSE}_{\text{Newton}} \gtrsim 10^{-10}$; on the other hand, the largest maximum error made by the quasi-Newton method (across all configurations) amounts at $10^{-5} \lesssim \text{MRDE}_{\text{Quasi-Newton}} \lesssim 10^{-1}$ and $10^{-5} \leq \text{MRSE}_{\text{Quasi-Newton}} \leq 10^{-4}$. For what concerns speed, Newton's method employs *tenths of seconds* to achieve convergence on each configuration while the quasi-Newton one always requires *tens of seconds* (specifically, almost thirty seconds for each considered configuration). The results above indicate that the fastest and most accurate method is systematically Newton's one, a result suggesting that the information encoded into the Hessian matrix cannot be ignored without consequences on the quality of the solution. The fixed-point algorithm, instead, stops after seconds but is affected by errors whose order of magnitude systematically amounts at $\text{MRDE}_{\text{fixed-point}} \simeq 10^2$ and $1 \lesssim \text{MRSE}_{\text{fixed-point}} \lesssim 10^2$.

We also explicitly notice that the MADE basically coincides with the MRDE for all considered configurations, meaning that the largest error made by the algorithms considered here to solve the UECM affects the nodes with the lowest degree (i.e. equal to one). On the other hand, strengths are affected by a larger absolute error (i.e. the MASE, defined as $\text{MASE} = \max_i\{|s_i^* - \langle s_i \rangle|\}_{i=1}^N$) than the degrees: if we calculate the MRSE, however, we realize that the largest errors affect very large strengths - hence being perfectly acceptable. For example, let us consider the WTW in 1993: the MASE affecting the quasi-Newton method amounts at 0.1 but, as the MRSE reveals, it affects a strength of the order of 10^3.

Lastly, differences in the speed of convergence of the two methods discussed in this section, caused by the choice of a particular set of initial conditions, are observable: the 'uniform' prescription outperforms the

				Newton				Quasi-Newton			
	N	L	c	MRDE	MASE	MRSE	Time (s)	MRDE	MASE	MRSE	Time (s)
WTW 90	169	7991	$\simeq 0.3$	$\simeq 2.6 \cdot 10^{-10}$	$\simeq 5 \cdot 10^{-6}$	$\simeq 2 \cdot 10^{-10}$	$\simeq 0.4$	$\simeq 7.8 \cdot 10^{-4}$	$\simeq 2 \cdot 10^{-1}$	$\simeq 3 \cdot 10^{-4}$	$\simeq 25$
WTW 91	184	8712	$\simeq 0.3$	$\simeq 2.1 \cdot 10^{-10}$	$\simeq 1 \cdot 10^{-10}$	$\simeq 1.2 \cdot 10^{-10}$	$\simeq 0.5$	$\simeq 9.2 \cdot 10^{-5}$	$\simeq 7 \cdot 10^{-2}$	$\simeq 6.6 \cdot 10^{-5}$	$\simeq 28$
WTW 92	185	8928	$\simeq 0.3$	$\simeq 1.2 \cdot 10^{-10}$	$\simeq 7 \cdot 10^{-7}$	$\simeq 1.3 \cdot 10^{-10}$	$\simeq 0.5$	$\simeq 7 \cdot 10^{-2}$	$\simeq 2 \cdot 10^{-4}$	$\simeq 1.3 \cdot 10^{-4}$	$\simeq 28$
WTW 93	187	9220	$\simeq 0.3$	$\simeq 3.4 \cdot 10^{-6}$	$\simeq 2 \cdot 10^{-10}$	$\simeq 2.2 \cdot 10^{-10}$	$\simeq 0.5$	$\simeq 1.4 \cdot 10^{-4}$	$\simeq 1 \cdot 10^{-1}$	$\simeq 7.6 \cdot 10^{-5}$	$\simeq 28$
WTW 94	187	9437	$\simeq 0.3$	$\simeq 1.8 \cdot 10^{-10}$	$\simeq 2 \cdot 10^{-10}$	$\simeq 1.8 \cdot 10^{-10}$	$\simeq 0.7$	$\simeq 7 \cdot 10^{-5}$	$\simeq 2 \cdot 10^{-1}$	$\simeq 1.5 \cdot 10^{-4}$	$\simeq 28$
WTW 95	187	9578	$\simeq 0.3$	$\simeq 2.8 \cdot 10^{-10}$	$\simeq 3 \cdot 10^{-6}$	$\simeq 2.9 \cdot 10^{-10}$	$\simeq 0.6$	$\simeq 2.5 \cdot 10^{-4}$	$\simeq 3 \cdot 10^{-1}$	$\simeq 6.7 \cdot 10^{-5}$	$\simeq 28$
WTW 96	187	10002	$\simeq 0.3$	$\simeq 1.4 \cdot 10^{-5}$	$\simeq 1 \cdot 10^{-10}$	$\simeq 1.1 \cdot 10^{-10}$	$\simeq 0.7$	$\simeq 1.7 \cdot 10^{-5}$	$\simeq 2 \cdot 10^{-1}$	$\simeq 3 \cdot 10^{-5}$	$\simeq 28$
WTW 97	187	10251	$\simeq 0.3$	$\simeq 1.1 \cdot 10^{-5}$	$\simeq 4 \cdot 10^{-10}$	$\simeq 6.7 \cdot 10^{-10}$	$\simeq 0.7$	$\simeq 4.4 \cdot 10^{-5}$	$\simeq 8 \cdot 10^{-2}$	$\simeq 1.6 \cdot 10^{-4}$	$\simeq 28$
WTW 98	187	10254	$\simeq 0.3$	$\simeq 1 \cdot 10^{-5}$	$\simeq 3 \cdot 10^{-10}$	$\simeq 4 \cdot 10^{-10}$	$\simeq 0.6$	$\simeq 1.7 \cdot 10^{-4}$	$\simeq 8 \cdot 10^{-2}$	$\simeq 5.3 \cdot 10^{-5}$	$\simeq 28$
WTW 99	187	10252	$\simeq 0.3$	$\simeq 4.7 \cdot 10^{-10}$	$\simeq 1 \cdot 10^{-10}$	$\simeq 8 \cdot 10^{-10}$	$\simeq 0.7$	$\simeq 1.6 \cdot 10^{-4}$	$\simeq 7 \cdot 10^{-2}$	$\simeq 6.2 \cdot 10^{-5}$	$\simeq 28$
WTW 00	187	10252	$\simeq 0.3$	$\simeq 5 \cdot 10^{-10}$	$\simeq 2 \cdot 10^{-10}$	$\simeq 2.4 \cdot 10^{-10}$	$\simeq 0.7$	$\simeq 1.5 \cdot 10^{-4}$	$\simeq 9 \cdot 10^{-2}$	$\simeq 5.4 \cdot 10^{-5}$	$\simeq 29$

Table 2: Performance of Newton's and the quasi-Newton method to solve the reduced system of equations defining the UECM, on a set of real-world WUNs (of which basic statistics as the total number of nodes, N, the total number of links, L, and the connectance, $c = 2L/N(N-1)$, are provided). While Newton's method stops because the condition $||\nabla \mathscr{L}(\vec{\theta})||_2 \leq 10^{-8}$ is satisfied, the quasi-Newton one always reaches the limit of 10000 steps. The results on accuracy and speed clearly indicate that Newton's method outperforms the quasi-Newton one. Only the results corresponding to the best choice of initial conditions are reported. The results of the fixed-point recipe are not shown.

Algorithm 2 Sampling the UECM ensemble

```
1: for $m = 1 \dots |E|$ do
2:     $\mathbf{W} = \mathbf{0}$;
3:     for $i = 1 \dots N$ do
4:         for $j = 1 \dots N$ and $j < i$ do
5:             if RandomUniform$[0, 1] \leq p_{ij}^{\text{UECM}}$ then
6:                 $w_{ij} = w_{ji} = $ RandomGeometric$[e^{-\beta_i - \beta_j}]$;
7:             else
8:                 $w_{ij} = w_{ji} = 0$;
9:             end if
10:        end for
11:    end for
12:    Ensemble$[m] = \mathbf{W}$;
13: end for
```

other ones.

Sampling the UECM

Finally, let us comment on the algorithm to sample the UECM ensemble and that can be compactly achieved by implementing a two-step procedure. Let us look back at the formal expression for the pair-specific probability distribution characterizing the UECM: it induces coefficients reading

$$\begin{cases} 1 - p_{ij}^{\text{UECM}}, & w = 0 \\ p_{ij}^{\text{UECM}}(1 - e^{-\beta_i - \beta_j}), & w = 1 \\ p_{ij}^{\text{UECM}}(e^{-\beta_i - \beta_j})(1 - e^{-\beta_i - \beta_j}), & w = 2 \\ p_{ij}^{\text{UECM}}(e^{-\beta_i - \beta_j})^2(1 - e^{-\beta_i - \beta_j}), & w = 3 \\ \qquad\qquad\vdots \end{cases} \tag{2.47}$$

in turn suggesting that, for a specific pair of vertices i, j (with $i < j$), the appearance of the first link is ruled by a Bernoulli distribution with probability p_{ij}^{UECM} while the remaining $(w - 1)$ ones can be drawn from a geometric distribution whose parameter reads $e^{-\beta_i - \beta_j}$; in other words, the weight $(w - 1)$ is drawn *conditionally* on the presence of a connection between the two considered nodes. The computational complexity of the sampling process is, again, $O(N^2)$. The pseudo-code for explicitly sampling the UECM ensemble is summed up by Algorithm 2.

2.6.3 Weighted directed graphs with given strengths and degrees (DECM)

Let us now extend the 'mixed' model introduced in the previous section to the case of directed networks. Constraints are, now, represented by four sequences of values, i.e. $\{k_i^{out}\}_{i=1}^N$, $\{k_i^{in}\}_{i=1}^N$, $\{s_i^{out}\}_{i=1}^N$, $\{s_i^{in}\}_{i=1}^N$ where the generic out-degree and in-degree are, respectively, defined as $k_i^{out}(\mathbf{A}) = \sum_{j(\neq i)=1}^N a_{ij}$ and $k_i^{in}(\mathbf{A}) = \sum_{j(\neq i)=1}^N a_{ji}$ and analogously for the generic out-strength and in-strength, reading $s_i^{out}(\mathbf{W}) = \sum_{j(\neq i)=1}^N w_{ij}$ and $s_i^{in}(\mathbf{W}) = \sum_{j(\neq i)=1}^N w_{ji}$. Consistency requires that $\mathbf{A} \equiv \Theta[\mathbf{W}]$ as for the UECM case. This model is known as *Directed Enhanced Configuration Model* (DECM) and its Hamiltonian reads

$$\mathcal{H}_{\text{DECM}}(\mathbf{W}, \vec{\alpha}, \vec{\beta}, \vec{\gamma}, \vec{\delta}) = \sum_{i=1}^N [\alpha_i k_i^{out}(\mathbf{A}) + \beta_i k_i^{in}(\mathbf{A}) + \gamma_i s_i^{out}(\mathbf{W}) + \delta_i s_i^{in}(\mathbf{W})] \tag{2.48}$$

in turn, inducing the directed counterpart of the UECM distribution, i.e.

$$Q_{\text{DECM}}(\mathbf{W}|\vec{\alpha}, \vec{\beta}, \vec{\gamma}, \vec{\delta}) = \prod_{i=1}^N \prod_{\substack{j=1 \\ (j\neq i)}}^N q_{ij}(w_{ij}) \tag{2.49}$$

with

$$q_{ij}(w) = \begin{cases} 1 - p_{ij}^{\text{DECM}} & w = 0 \\ p_{ij}^{\text{DECM}}(e^{-\gamma_i - \delta_j})^{w-1}(1 - e^{-\gamma_i - \delta_j}) & w > 0 \end{cases} \tag{2.50}$$

for any two nodes i and j such that $i \neq j$ and

$$p_{ij}^{\text{DECM}} = \frac{e^{-\alpha_i - \beta_j - \gamma_i - \delta_j}}{1 - e^{-\gamma_i - \delta_j} + e^{-\alpha_i - \beta_j - \gamma_i - \delta_j}}, \quad \forall\, i \neq j. \tag{2.51}$$

As for the undirected case, weights are required to assume only (non-negative) integer values (i.e. $w_{ij} \in [0, +\infty), \forall i \neq j$). Hence, the canonical ensemble is constituted by the weighted configurations with N nodes and a number of (directed) links ranging between zero and the maximum value $N(N-1)$.

The argument of the problem (2.14) for the specific network $\mathbf{W}^*$ becomes

$$\begin{aligned}\mathscr{L}_{\text{DECM}}(\vec{\alpha}, \vec{\beta}, \vec{\gamma}, \vec{\delta}) = &-\sum_{i=1}^{N}[\alpha_i k_i^{out}(\mathbf{A}^*) + \beta_i k_i^{in}(\mathbf{A}^*) \\ &+ \gamma_i s_i^{out}(\mathbf{W}^*) + \delta_i s_i^{in}(\mathbf{W}^*)] \\ &-\sum_{i=1}^{N}\sum_{\substack{j=1 \\ (j\neq i)}}^{N} \ln z_{ij}\end{aligned} \tag{2.52}$$

where $z_{ij} = \left[1 + e^{-\alpha_i - \beta_j}\left(\frac{e^{-\gamma_i-\delta_j}}{1-e^{-\gamma_i-\delta_j}}\right)\right]$, $\forall\, i \neq j$ and whose first-order, optimality conditions read

$$\begin{aligned}
\nabla_{\alpha_i}\mathscr{L}_{\text{DECM}} &= -k_i^{out}(\mathbf{A}^*) + \sum_{\substack{j=1 \\ (j\neq i)}}^{N} p_{ij}^{\text{DECM}} = -k_i^{out}(\mathbf{A}^*) + \langle k_i^{out}\rangle = 0 \\
\nabla_{\beta_i}\mathscr{L}_{\text{DECM}} &= -k_i^{in}(\mathbf{A}^*) + \sum_{\substack{j=1 \\ (j\neq i)}}^{N} p_{ji}^{\text{DECM}} = -k_i^{in}(\mathbf{A}^*) + \langle k_i^{in}\rangle = 0 \\
\nabla_{\gamma_i}\mathscr{L}_{\text{DECM}} &= -s_i^{out}(\mathbf{W}^*) + \sum_{\substack{j=1 \\ (j\neq i)}}^{N} \frac{p_{ij}^{\text{DECM}}}{1-e^{-\gamma_i-\delta_j}} = -s_i^{out}(\mathbf{W}^*) + \langle s_i^{out}\rangle = 0 \\
\nabla_{\delta_i}\mathscr{L}_{\text{DECM}} &= -s_i^{in}(\mathbf{W}^*) + \sum_{\substack{j=1 \\ (j\neq i)}}^{N} \frac{p_{ji}^{\text{DECM}}}{1-e^{-\gamma_j-\delta_i}} = -s_i^{in}(\mathbf{W}^*) + \langle s_i^{in}\rangle = 0
\end{aligned} \tag{2.53}$$

with $i = 1 \dots N$.

Resolution of the DECM

Newton's and the quasi-Newton methods can be easily implemented via the recipe defined in eq. (2.18) (see Appendix A.1 for the definition of the DECM Hessian).

As for the UECM, the fixed-point recipe for solving the DECM first-order, optimality conditions transforms the following set of consistency equations

$$\begin{aligned}
\alpha_i &= -\ln\left[\frac{k_i^{out}(\mathbf{A}^*)}{\sum_{\substack{j=1\\(j\neq i)}}^{N}\left(\frac{e^{-\beta_j-\gamma_i-\delta_j}}{1-e^{-\gamma_i-\delta_j}+e^{-\alpha_i-\beta_j-\gamma_i-\delta_j}}\right)}\right], \quad i=1\dots N\\
\beta_i &= -\ln\left[\frac{k_i^{in}(\mathbf{A}^*)}{\sum_{\substack{j=1\\(j\neq i)}}^{N}\left(\frac{e^{-\alpha_j-\gamma_j-\delta_i}}{1-e^{-\gamma_j-\delta_i}+e^{-\alpha_j-\beta_i-\gamma_j-\delta_i}}\right)}\right], \quad i=1\dots N\\
\gamma_i &= -\ln\left[\frac{s_i^{out}(\mathbf{W}^*)}{\sum_{\substack{j=1\\(j\neq i)}}^{N}\left(\frac{e^{-\alpha_i-\beta_j-\delta_j}}{(1-e^{-\gamma_i-\delta_j})(1-e^{-\gamma_i-\delta_j}+e^{-\alpha_i-\beta_j-\gamma_i-\delta_j})}\right)}\right], \quad i=1\dots N\\
\delta_i &= -\ln\left[\frac{s_i^{in}(\mathbf{W}^*)}{\sum_{\substack{j=1\\(j\neq i)}}^{N}\left(\frac{e^{-\alpha_j-\beta_i-\gamma_j}}{(1-e^{-\gamma_j-\delta_i})(1-e^{-\gamma_j-\delta_i}+e^{-\alpha_j-\beta_i-\gamma_j-\delta_i})}\right)}\right], \quad i=1\dots N
\end{aligned} \tag{2.54}$$

into the usual iterative fashion, by considering the parameters at the left hand side and at the right hand side, respectively at the n-th and at the $(n-1)$-th iteration. The reduced version of such a recipe would assign the same set of values $(\alpha, \beta, \gamma, \delta)$ to the nodes for which the quantities $(k^{out}, k^{in}, s^{out}, s^{in})$ have the same value: however, the larger heterogeneity of the strengths causes the DECM to be much less reducible than the UBCM model (previously discussed in the present chapter).

The three different sets of initial conditions that have been considered generalize the UECM ones: in particular, the first set of values reads $\alpha_i^{(0)} = -\ln\left[\frac{k_i^{out}(\mathbf{A}^*)}{\sqrt{L}}\right]$, $i = 1\dots N$, $\beta_i^{(0)} = -\ln\left[\frac{k_i^{in}(\mathbf{A}^*)}{\sqrt{L}}\right]$, $i = 1\dots N$, $\gamma_i^{(0)} = -\ln\left[\frac{s_i^{out}(\mathbf{W}^*)}{\sqrt{W}}\right]$, $i = 1\dots N$ and $\delta_i^{(0)} = -\ln\left[\frac{s_i^{in}(\mathbf{W}^*)}{\sqrt{W}}\right]$, $i = 1\dots N$; the second set of initial conditions can be obtained by simply replacing L with N; the third recipe, as usual, prescribes to randomly draw the value of each parameter from the uniform distribution defined on the unit interval.

Performance testing

The accuracy of each algorithm in reproducing the constraints defining the DECM has been quantified via the *maximum relative error* metrics, now reading

$$\text{MRDE} = \max_i \left\{ \frac{|k_i^* - \langle k_i \rangle|}{k_i}, \frac{|h_i^* - \langle h_i \rangle|}{h_i} \right\}_{i=1}^N \tag{2.55}$$

$$\text{MRSE} = \max_i \left\{ \frac{|s_i^* - \langle s_i \rangle|}{s_i}, \frac{|t_i^* - \langle t_i \rangle|}{t_i} \right\}_{i=1}^N \tag{2.56}$$

(the acronyms standing for for Maximum Relative Degree Error and Maximum Relative Strength Error) where we have defined $k^{out} \equiv k$, $k^{in} \equiv h$, $s^{out} \equiv s$ and $s^{in} \equiv t$ in order to simplify the formalism.

The three different 'stop criteria' we have adopted are the same ones we have considered for both the binary and the undirected, 'mixed' model, i.e. the condition on the Euclidean norm of the gradient of the likelihood function, i.e. $||\nabla \mathscr{L}(\vec{\theta})||_2 \leq 10^{-8}$), the condition on the Euclidean norm of the vector of differences between the values of the parameters at subsequent iterations (i.e. $||\Delta\vec{\theta}||_2 \leq 10^{-8}$) and the condition on the maximum number of iterations (i.e. after 10000 steps, any of the three algorithms stops).

Results

The performance of the three algorithms to solve the system of equations defining the DECM has been tested on a bunch of real-world networks: in particular, we have considered the Electronic Italian Interbank Market (e-MID) during the decade 2000-2010[5] [61]. Before commenting on the results of our numerical exercises, let us, first, describe how the latter ones have been carried out.

The results about the performance of our three algorithms are reported in Table 3. Overall, Newton's method performs very satisfactorily, being accurate, fast and scalable; the quasi-Newton method is accurate as well although (in some cases, much) slower. The fixed-point recipe, instead, performs very poorly, as for the undirected case. Moreover, while Newton's method stops because the condition on the norm of the likelihood is satisfied, both the quasi-Newton and the fixed-point algorithms are always found to satisfy the limit condition on the number of steps (i.e. they run for 10000 steps and, then, stop).

For what concerns accuracy, the largest maximum error made by Newton's method (across all configurations) amounts at $10^{-14} \lesssim \text{MRDE}_{\text{Newton}} \lesssim 10^{-7}$ and $10^{-12} \lesssim \text{MRSE}_{\text{Newton}} \lesssim 10^{-5}$; on the other hand, the largest maximum error made by the quasi-Newton method (across all configurations) amounts at $10^{-8} \lesssim \text{MRDE}_{\text{Quasi-Newton}} \lesssim 10^{-7}$ and $10^{-4} \lesssim \text{MRSE}_{\text{Quasi-Newton}} \lesssim 10^{-3}$. For what concerns speed, Newton's method employs tens of seconds to achieve convergence on each configuration; the time required by the quasi-Newton method is of the same order of magnitude, although it is systematically larger than the time required by Newton's one. Overall, these results indicate that the fastest and most accurate method is Newton's one. As in the undirected case, the fixed-point algorithm, instead, stops after seconds but is affected by errors whose order of magnitude systematically amounts at $10 \lesssim \text{MRDE}_{\text{fixed-point}} \lesssim 10^2$ and $1 \lesssim \text{MRSE}_{\text{fixed-point}} \lesssim 10^2$.

As for the UECM, the MADE basically coincides with the MRDE, for all considered configurations, while strengths are affected by a larger ab-

[5]Since e-MID weights are real numbers, we have rounded them to the nearest integer value, before running the DECM.

				Newton				Quasi-Newton			
	N	L	c	MRDE	MASE	MRSE	Time (s)	MRDE	MASE	MRSE	Time (s)
e-MID 00	196	10618	$\simeq 0.28$	$\simeq 1.4 \cdot 10^{-10}$	$\simeq 5 \cdot 10^{-9}$	$\simeq 1.5 \cdot 10^{-10}$	$\simeq 0.5$	$\simeq 1.7 \cdot 10^{-7}$	$\simeq 3 \cdot 10^{-1}$	$\simeq 5.4 \cdot 10^{-7}$	$\simeq 0.1$
e-MID 01	185	8951	$\simeq 0.26$	$\simeq 1.4 \cdot 10^{-11}$	$\simeq 6 \cdot 10^{-9}$	$\simeq 2 \cdot 10^{-10}$	$\simeq 0.4$	$\simeq 1.4 \cdot 10^{-7}$	$\simeq 10$	$\simeq 7 \cdot 10^{-5}$	$\simeq 0.2$
e-MID 02	177	7252	$\simeq 0.23$	$\simeq 1.4 \cdot 10^{-15}$	$\simeq \cdot 10^{-4}$	$\simeq 1 \cdot 10^{-5}$	$\simeq 0.5$	$\simeq 9.5 \cdot 10^{-8}$	$\simeq 6 \cdot 10^{-1}$	$\simeq 7.4 \cdot 10^{-6}$	$\simeq 0.1$
e-MID 03	179	6814	$\simeq 0.21$	$\simeq 1.6 \cdot 10^{-10}$	$\simeq 2 \cdot 10^{-5}$	$\simeq 4.4 \cdot 10^{-10}$	$\simeq 0.9$	$\simeq 9.6 \cdot 10^{-8}$	$\simeq 50$	$\simeq 1.1 \cdot 10^{-3}$	$\simeq 0.2$
e-MID 04	180	6136	$\simeq 0.19$	$\simeq 6.5 \cdot 10^{-13}$	$\simeq 9 \cdot 10^{-7}$	$\simeq 3.4 \cdot 10^{-12}$	$\simeq 0.9$	$\simeq 1 \cdot 10^{-7}$	$\simeq 700$	$\simeq 4.2 \cdot 10^{-3}$	$\simeq 0.2$
e-MID 05	176	6203	$\simeq 0.2$	$\simeq 3 \cdot 10^{-12}$	$\simeq 1 \cdot 10^{-5}$	$\simeq 9.4 \cdot 10^{-11}$	$\simeq 1.2$	$\simeq 4.8 \cdot 10^{-8}$	$\simeq 300$	$\simeq 2.4 \cdot 10^{-3}$	$\simeq 0.3$
e-MID 06	177	6132	$\simeq 0.19$	$\simeq 1.5 \cdot 10^{-14}$	$\simeq 2 \cdot 10^{-7}$	$\simeq 1.8 \cdot 10^{-11}$	$\simeq 0.9$	$\simeq 5 \cdot 10^{-8}$	$\simeq 60$	$\simeq 2.5 \cdot 10^{-3}$	$\simeq 0.2$
e-MID 07	178	6330	$\simeq 0.2$	$\simeq 8.4 \cdot 10^{-15}$	$\simeq 1 \cdot 10^{-4}$	$\simeq 2.5 \cdot 10^{-6}$	$\simeq 0.7$	$\simeq 1.8 \cdot 10^{-8}$	$\simeq 3$	$\simeq 7.6 \cdot 10^{-5}$	$\simeq 0.1$
e-MID 08	173	4767	$\simeq 0.16$	$\simeq 2.3 \cdot 10^{-9}$	$\simeq 2 \cdot 10^{-9}$	$\simeq 7.7 \cdot 10^{-10}$	$\simeq 0.6$	$\simeq 1.8 \cdot 10^{-8}$	$\simeq 20$	$\simeq 1.2 \cdot 10^{-3}$	$\simeq 0.3$
e-MID 09	156	2961	$\simeq 0.12$	$\simeq 2.6 \cdot 10^{-15}$	$\simeq 1 \cdot 10^{-7}$	$\simeq 4.9 \cdot 10^{-12}$	$\simeq 0.6$	$\simeq 1.7 \cdot 10^{-8}$	$\simeq 1$	$\simeq 9 \cdot 10^{-5}$	$\simeq 0.1$
e-MID 10	135	2743	$\simeq 0.15$	$\simeq 6.3 \cdot 10^{-8}$	$\simeq 3 \cdot 10^{-6}$	$\simeq 5.9 \cdot 10^{-10}$	$\simeq 0.7$	$\simeq 1.6 \cdot 10^{-8}$	$\simeq 4$	$\simeq 5.2 \cdot 10^{-5}$	$\simeq 0.1$

Table 3: Performance of Newton's and the quasi-Newton method to solve the reduced system of equations defining the DECM, on a set of real-world WDNs (of which basic statistics as the total number of nodes, N, the total number of links, L, and the connectance, $c = L/N(N-1)$, are provided). While Newton's method stops because the condition $||\nabla \mathscr{L}(\vec{\theta})||_2 \leq 10^{-8}$ is satisfied, the quasi-Newton one always reaches the limit of 10000 steps. The results on accuracy and speed clearly indicate that Newton's method outperforms the quasi-Newton one. Only the results corresponding to the best choice of initial conditions are reported. The results of the fixed-point recipe are not shown.

solute error than the degrees: still, upon calculating the MRSE, we realize that the largest errors affect very large strengths - hence being perfectly acceptable.

Lastly, differences in the speed of convergence of the two methods discussed in this section, caused by the choice of a particular set of initial conditions, are observable: the 'uniform' prescription outperforms the other ones.

Sampling the DECM

Finally, let us comment on the algorithm to sample the DECM ensemble: as for the UECM, it can be compactly achieved by implementing the directed counterpart of the two-step procedure described above. Given a specific pair of vertices i, j (with $i \neq j$), the first link can be drawn by sampling a Bernoulli distribution with probability p_{ij}^{DECM} while the remaining $(w-1)$ ones can be drawn from a geometric distribution whose parameter reads $e^{-\gamma_i-\delta_j}$. The computational complexity of the sampling process is, again, $O(N^2)$ and the pseudo-code for explicitly sampling the DECM ensemble is summed up by Algorithm 3.

2.6.4 Two-step models for undirected and directed networks

The need of considering network models defined in a two-step fashion arises from a number of considerations. First, recipes like the UECM and the DECM are, generally speaking, difficult to solve; as we have already observed, only Newton's method performs in a satisfactory way, both for what concerns accuracy and speed: hence, easier-to-solve recipes are welcome. Second, the amount of information concerning binary and weighted quantities is often asymmetric: as it has been pointed out in [62], information concerning a given network structure ranges from the knowledge of just a single, aggregated piece of information (e.g. the link density) to that of entire subgraphs.

Indeed, models exist that take as input any binary, either probabilistic or deterministic, network model - i.e. any $P(\mathbf{A})$ - while placing link

Algorithm 3 Sampling the DECM ensemble

```
1: for $m = 1 \ldots |E|$ do
2:     $\mathbf{W} = \mathbf{0}$;
3:     for $i = 1 \ldots N$ do
4:         for $j = 1 \ldots N$ and $j \neq i$ do
5:             if RandomUniform$[0, 1] \leq p_{i\alpha}^{\text{DECM}}$ then
6:                 $w_{ij} = $ RandomGeometric$[e^{-\gamma_i - \delta_j}]$;
7:             else
8:                 $w_{ij} = 0$;
9:             end if
10:        end for
11:    end for
12:    Ensemble$[m] = \mathbf{W}$;
13: end for
```

weights optimally, *conditionally* on the input configurations [26, 62]. For the sake of illustration, let us focus on undirected networks and consider the conditional reconstruction method (hereby, CReM) induced by the Hamiltonian

$$\mathcal{H}_{\text{CReM}}(\mathbf{W}, \vec{\theta}) = \sum_{i=1}^{N} \theta_i s_i(\mathbf{A}); \tag{2.57}$$

it leads to a conditional probability distribution reading

$$\mathbf{Q}(\mathbf{W}|\mathbf{A}) = \prod_{i=1}^{N} \prod_{\substack{j=1 \\ (j<i)}}^{N} q_{ij}(w_{ij}|a_{ij}) \tag{2.58}$$

where, for consistency, $q_{ij}(w_{ij} = 0|a_{ij} = 0) = 1$ and $q_{ij}(w_{ij} = 0|a_{ij} = 1) = 0$. The meaning of these relationships is the following: given any two nodes i and j, the absence of a link, i.e. $a_{ij} = 0$, admits the only possibility $w_{ij} = 0$; on the other hand, the presence of a link, i.e. $a_{ij} = 1$, rules out the possibility that a null weight among the same vertices is observed.

In general, the functional form of $q_{ij}(w_{ij}|a_{ij} = 1)$ depends on the domain of the weights. In all cases considered in [26, 62], weights are assumed to be continuous; since the continuous distribution that maximizes Shannon entropy, while constrained to reproduce first-order moments, is the exponential one, the following functional form

$$q_{ij}(w_{ij}|a_{ij}=1)=\begin{cases}(\theta_i+\theta_j)e^{-(\theta_i+\theta_j)w} & w>0\\ 0 & w\leq 0\end{cases} \tag{2.59}$$

(for any undirected pair of nodes) remains naturally induced. As shown in [62], the problem 2.14 has to be slightly generalized; still, its argument for the specific network $\mathbf{W}^*$ becomes

$$\mathcal{G}_{\text{CReM}}=-\sum_{i=1}^{N}\theta_i s_i(\mathbf{W}^*)+\sum_{i=1}^{N}\sum_{\substack{j=1\\(j<i)}}^{N}f_{ij}\log\left[\theta_i+\theta_j\right] \tag{2.60}$$

where the quantity $f_{ij}=\sum_{\mathbf{A}}P(\mathbf{A})a_{ij}$ represents the expected value of a_{ij} over the ensemble of (binary) configurations defining the binary model taken as input (i.e. the marginal probability that an edge exists between nodes i and j). It follows that the CReM first-order, optimality conditions read

$$\nabla_{\theta_i}\mathscr{L}_{\text{CReM}}=-s_i(\mathbf{W}^*)+\sum_{\substack{j=1\\(j\neq i)}}^{N}\frac{f_{ij}}{\theta_i+\theta_j}=-s_i(\mathbf{W}^*)+\langle s_i\rangle=0 \tag{2.61}$$

with $i=1\ldots N$.

Resolution of the CReM

Newton's and the quasi-Newton method can still be implemented via the recipe defined in eq. (2.18) (see Appendix A.1 for the definition of the CReM Hessian).

As for the UECM and the DECM, the fixed-point recipe for solving the system of equations embodying the CReM transforms the set of consistency equations

$$\theta_i = \left[\frac{s_i(\mathbf{W}^*)}{\sum_{\substack{j=1 \\ (j \neq i)}}^N \left(\frac{f_{ij}}{1+\theta_j/\theta_i} \right)} \right]^{-1}, \quad i = 1 \dots N \tag{2.62}$$

into an iterative recipe of the usual form, i.e. by considering the parameters at the left hand side and at the right hand side, respectively at the n-th and at the $(n-1)$-th iteration. Although a reduced recipe can, in principle, be defined, an analogous observation to the one concerning the UECM and the DECM holds: the mathematical nature of the strengths (now, real numbers) increases their heterogeneity, in turn causing the CReM algorithm to be reducible even less than the 'mixed' models defined by discrete weights.

The initialization of the iterative recipe for solving the CReM has been implemented in the usual threefold way. The first set of initial values reads $\theta_i^{(0)} = -\ln\left[\frac{s_i(\mathbf{W}^*)}{\sqrt{2W}}\right]$, $i = 1 \dots N$; the second one is a variant of the position above, reading $\theta_i^{(0)} = -\ln\left[\frac{s_i(\mathbf{W}^*)}{\sqrt{N}}\right]$; the third one, instead, prescribes to randomly draw the value of each parameter from the uniform distribution defined on the unit interval, i.e. $\theta_i^{(0)} \sim \mathrm{U}(0,1), \forall\, i$.

When considering directed networks, instead, the conditional probability distribution defining the CReM reads

$$q_{ij}(w_{ij}|a_{ij}=1) = \begin{cases} (\alpha_i + \beta_j) e^{-(\alpha_i+\beta_j)w} & w > 0 \\ 0 & w \leq 0 \end{cases} \tag{2.63}$$

for any two nodes i and j such that $i \neq j$; the set of equations (2.62) can be generalized as follows

$$\alpha_i = \left[\frac{s_i^{out}(\mathbf{W}^*)}{\sum_{\substack{j=1\\(j\neq i)}}^N \left(\frac{f_{ij}}{1+\beta_j/\alpha_i}\right)}\right]^{-1}, \quad i = 1\dots N$$
$$\beta_i = \left[\frac{s_i^{in}(\mathbf{W}^*)}{\sum_{\substack{j=1\\(j\neq i)}}^N \left(\frac{f_{ji}}{1+\alpha_j/\beta_i}\right)}\right]^{-1}, \quad i = 1\dots N \tag{2.64}$$

and analogously for the sets of values initializing them.

Rescaling the CReM algorithm

Although the equations defining the CReM algorithm cannot be effectively reduced, they can be opportunely *rescaled*. To this aim, let us consider directed configurations and the system

$$\sum_{\substack{j=1\\j(\neq i)}}^N \frac{f_{ij}}{\alpha_i(\kappa)+\beta_j(\kappa)} = \frac{s_i^{out}(\mathbf{W}^*)}{\kappa}, \quad i = 1\dots N \tag{2.65}$$
$$\sum_{\substack{j=1\\j(\neq i)}}^N \frac{f_{ji}}{\alpha_j(\kappa)+\beta_i(\kappa)} = \frac{s_i^{in}(\mathbf{W}^*)}{\kappa}, \quad i = 1\dots N \tag{2.66}$$

where the sufficient statistics has been divided by an opportunely defined factor (in this case, κ) and the symbols $\alpha_i(\kappa)$, $\alpha_j(\kappa)$, $\beta_i(\kappa)$ and $\beta_j(\kappa)$ stress that the solution we are searching for is a function of the parameter κ itself. In fact, a solution of the system above reads

$$\alpha_i^*(\kappa) = \kappa \cdot \alpha_i^*, \quad i = 1\dots N \tag{2.67}$$
$$\beta_i^*(\kappa) = \kappa \cdot \beta_i^*, \quad i = 1\dots N \tag{2.68}$$

as it can be proven upon substituting it back and noticing that $\{\alpha_i^*\}_{i=1}^N$ and $\{\beta_i^*\}_{i=1}^N$ are solutions of the same system of equations with $\kappa = 1$. As our likelihood maximization problem admits a unique, global maximum, the prescription above allows us to easily identify it. Rescaling will be tested in order to find out if our algorithms are enhanced by it under any respect (e.g. accuracy or speed).

Performance testing

Before commenting on the performance of the three algorithms in solving the system of equations defining the CReM, let us stress once more that the formulas presented so far are perfectly general, working for any binary recipe one may want to employ. In what follows, we will test the CReM by posing $f_{ij} \equiv p_{ij}^{\text{UBCM}}$ and $f_{ij} \equiv p_{ij}^{\text{DBCM}}$.

As for the discrete 'mixed' models, the accuracy of each algorithm in reproducing the constraints defining the CReM has been quantified via the Maximum Relative Degree Error and the Maximum Relative Strength Error metrics, whose definition is provided by eqs. (2.45), (2.46) and (2.55), (2.56) for the undirected and the directed case, respectively. Analogously, the three 'stop criteria' for each algorithm are the same ones that we have adopted for the other models (and consist in a condition on the Euclidean norm of the gradient of the likelihood function, i.e. $||\nabla\mathscr{L}(\vec{\theta})||_2 \leq 10^{-8}$, a condition on the Euclidean norm of the vector of differences between the values of the parameters at subsequent iterations, i.e. $||\Delta\vec{\theta}||_2 \leq 10^{-8}$, and a condition on the maximum number of iterations, i.e. 10.000 steps).

Results

To test the effectiveness of our algorithms in solving the CReM on undirected networks we have considered the synaptic network of the worm *C. Elegans* [51] and the eight daily snapshots of the Bitcoin Lightning Network [55]; the directed version of the CReM has, instead, been solved on the Electronic Italian Interbank Market (e-MID) during the decade 2000-2010 [61]. Before commenting on the results of our numerical exercises, let us, first, describe how the latter ones have been carried out.

The results about the performance of our three algorithms are reported in Table 4 and in Table 5. Let us start by commenting the results reported in Table 4 and concerning undirected networks. Generally speaking, Newton's method is the most accurate one (its largest maximum errors span intervals, across all configurations, that amount at $10^{-11} \lesssim \text{MRDE}_{\text{Newton}} \lesssim 10^{-8}$ and $10^{-5} \lesssim \text{MRSE}_{\text{Newton}} \lesssim 10^{-4}$) although it scales very badly with the size of the network on which it is tested (the amount of time, measured in seconds, required by it to achieve convergence spans an interval, across all configurations, that amounts at $0.08 \leq T_{\text{Newton}}^{\text{reduced}} \leq 1.188$).

The quasi-Newton method, on the other hand, is very accurate on the degrees (as already observed in the UBCM case) but not so accurate in reproducing the weighted constraints (its largest maximum errors span intervals, across all configurations, that amount at $\text{MRDE}_{\text{Quasi-Newton}} \simeq 10^{-7}$ and $10^{-6} \lesssim \text{MRSE}_{\text{Quasi-Newton}} \lesssim 6$). Moreover, it scales even worse than Newton's method with the size of the network on which it is tested (the amount of time, measured in seconds, required by it to achieve convergence spans an interval, across all configurations, that amounts at $5 \leq T_{\text{Quasi-Newton}} \leq 15.888$).

The performance of the fixed-point recipe is, somehow, intermediate between that of Newton's and that of the quasi-Newton method. For what concerns accuracy, it is more accurate in reproducing the binary constraints than in reproducing the weighted ones (its largest maximum errors span intervals, across all configurations, that amount at $\text{MRDE}_{\text{fixed-point}} \simeq 10^{-9}$ and $10^{-8} \lesssim \text{MRSE}_{\text{fixed-point}} \lesssim 10^{-1}$) although it outperforms Newton's method, sometimes. For what concerns scalability, the fixed-point method is the less sensitive one to the growing size of the considered configurations: hence, it is also the fastest one (the amount of time, measured in seconds, required by it to achieve convergence spans an interval, across all configurations, that amounts at $0.01 \leq T_{\text{fixed-point}} \leq 550$).

Moreover, while Newton's and the fixed-point method stop because the condition on the norm of the likelihood is satisfied, the quasi-Newton

	Newton				Quasi-Newton				Fixed-point			
	MRDE	MASE	MRSE	Time (s)	MRDE	MASE	MRSE	Time (s)	MRDE	MASE	MRSE	Time (s)
BLN$_1$	$\simeq 4 \cdot 10^{-8}$	$\simeq 10^{-9}$	$\simeq 10^{-4}$	$\simeq 0.08$	$\simeq 6 \cdot 10^{-8}$	$\simeq 10^{-4}$	$\simeq 10^{-6}$	$\simeq 5$	$\simeq 2 \cdot 10^{-9}$	$\simeq 10^{-2}$	$\simeq 10^{-1}$	$\simeq 0.01$
BLN$_2$	$\simeq 2 \cdot 10^{-8}$	$\simeq 10^{-8}$	$\simeq 10^{-5}$	$\simeq 3.2$	$\simeq 9 \cdot 10^{-8}$	$\simeq 10^{-4}$	$\simeq 10^{-1}$	$\simeq 100$	$\simeq 1 \cdot 10^{-9}$	$\simeq 10^{-2}$	$\simeq 10^{-1}$	$\simeq 0.73$
BLN$_3$	$\simeq 1 \cdot 10^{-8}$	$\simeq 10^{-9}$	$\simeq 10^{-4}$	$\simeq 14$	$\simeq 1 \cdot 10^{-7}$	$\simeq 10^{-4}$	$\simeq 10^{-2}$	$\simeq 388$	$\simeq 1 \cdot 10^{-9}$	$\simeq 10^{-7}$	$\simeq 10^{-6}$	$\simeq 11$
BLN$_4$	$\simeq 2 \cdot 10^{-8}$	$\simeq 10^{-9}$	$\simeq 10^{-4}$	$\simeq 71$	$\simeq 5 \cdot 10^{-8}$	$\simeq 10^{-2}$	$\simeq 3$	$\simeq 1538$	$\simeq 9 \cdot 10^{-10}$	$\simeq 20$	$\simeq 6 \cdot 10^{-1}$	$\simeq 1.3$
BLN$_5$	$\simeq 2 \cdot 10^{-9}$	$\simeq 10^{-9}$	$\simeq 10^{-4}$	$\simeq 200$	$\simeq 4 \cdot 10^{-7}$	$\simeq 10^{-1}$	$\simeq 6$	$\simeq 3633$	$\simeq 6 \cdot 10^{-10}$	$\simeq 10^{-7}$	$\simeq 10^{-8}$	$\simeq 5.7$
BLN$_6$	$\simeq 2 \cdot 10^{-9}$	$\simeq 10^{-9}$	$\simeq 10^{-4}$	$\simeq 382$	$\simeq 3 \cdot 10^{-8}$	$\simeq 10^{-2}$	$\simeq 3$	$\simeq 5980$	$\simeq 6 \cdot 10^{-10}$	$\simeq 10^{-3}$	$\simeq 10^{-3}$	$\simeq 550$
BLN$_7$	$\simeq 5 \cdot 10^{-8}$	$\simeq 10^{-9}$	$\simeq 10^{-4}$	$\simeq 648$	$\simeq 4 \cdot 10^{-7}$	$\simeq 10^{-2}$	$\simeq 2$	$\simeq 10177$	$\simeq 5 \cdot 10^{-10}$	$\simeq 10^{-2}$	$\simeq 10^{-1}$	$\simeq 36$
BLN$_8$	$\simeq 5 \cdot 10^{-12}$	$\simeq 10^{-10}$	$\simeq 10^{-6}$	$\simeq 1188$	$\simeq 3 \cdot 10^{-7}$	$\simeq 10^{-4}$	$\simeq 10^{-1}$	$\simeq 15888$	$\simeq 5 \cdot 10^{-10}$	$\simeq 10^{-2}$	$\simeq 10^{-1}$	$\simeq 70$

Table 4: Performance of Newton's, quasi-Newton and the fixed-point algorithm to solve the system of equations defining the (undirected version of the) CReM, on a set of real-world WUNs. While Newton's and the fixed-point method stop because the condition $||\nabla \mathscr{L}(\vec{\theta})||_2 \leq 10^{-8}$ is satisfied, the quasi-Newton one often reaches the limit of 10000 steps. The results on accuracy and speed indicate that Newton's and the fixed-point method compete, outperforming the quasi-Newton one. Only the results corresponding to the best choice of initial conditions are reported.

method is often found to satisfy the limit condition on the number of steps (i.e. it runs for 10000 steps and, then, stops).

Interestingly, the fact that the CReM cannot be reduced (at least not to a comparable extent with the one characterizing purely binary models) reveals a dependence on the network size of Newton's and of the quasi-Newton algorithms. The reason may lie in the evidence that both Newton's and the quasi-Newton method require (some proxy of) the Hessian matrix of the system of equations defining the CReM to update the value of the parameters: as already observed, the order of the latter - which is $O(N^2)$ for Newton's method and $O(N)$ for the quasi-Newton one - can make its calculation (very) time demanding.

Let us now move to comment on the performance of our algorithms when applied to solve the directed version of the CReM (see Table 5). Overall, all methods perform much better than in the undirected case, stopping because the condition on the norm of the likelihood is satisfied.

In fact, all of them are very accurate in reproducing the purely binary constraints, their largest maximum errors spanning intervals, across all configurations, that amount at $10^{-12} \lesssim \text{MRDE}_{\text{Newton}} \lesssim 10^{-6}$, $10^{-14} \lesssim \text{MRDE}_{\text{Quasi-Newton}} \lesssim 10^{-6}$ and $10^{-15} \lesssim \text{MRDE}_{\text{fixed-point}} \lesssim 10^{-8}$; for what concerns the weighted constraints, instead, the two most accurate methods are Newton's and the quasi-Newton one, their largest maximum errors spanning intervals, across all configurations, that amount at $10^{-13} \lesssim \text{MRSE}_{\text{Newton}} \lesssim 10^{-7}$ and $10^{-6} \lesssim \text{MRSE}_{\text{Quasi-Newton}} \lesssim 10^{-3}$ (the fixed-point method performs worse than them, since $10^{-3} \lesssim \text{MRSE}_{\text{fixed-point}} \lesssim 10^{-1}$).

For what concerns speed, the amount of time, measured in seconds, required by Newton's, the quasi-Newton and the fixed-point algorithms to achieve convergence spans an interval, across all configurations, that amounts at $0.6 \leq T^{\text{reduced}}_{\text{Newton}} \leq 1$, $0.5 \leq T^{\text{reduced}}_{\text{Quasi-Newton}} \leq 1.2$ and $0.05 \leq T^{\text{reduced}}_{\text{fixed-point}} \leq 0.2$, respectively: hence, all methods are also very fast - the fixed-point one being systematically the fastest.

As already stressed above, the fact that the e-MID number of nodes remains approximately constant throughout the considered time interval masks the strong dependence of the performance of Newton's and the

			Newton			Quasi-Newton			Fixed-point		
	N	L	MRDE	MRSE	Time (s)	MRDE	MRSE	Time (s)	MRDE	MRSE	Time (s)
e-MID 00	196	10618	$\simeq 3 \cdot 10^{-7}$	$\simeq 2 \cdot 10^{-10}$	$\simeq 0.9$	$\simeq 5 \cdot 10^{-7}$	$\simeq 2 \cdot 10^{-6}$	$\simeq 0.9$	$\simeq 4 \cdot 10^{-14}$	$\simeq 8 \cdot 10^{-5}$	$\simeq 0.09$
e-MID 01	185	8951	$\simeq 3 \cdot 10^{-15}$	$\simeq 1 \cdot 10^{-10}$	$\simeq 0.9$	$\simeq 7 \cdot 10^{-10}$	$\simeq 5 \cdot 10^{-6}$	$\simeq 1$	$\simeq 7 \cdot 10^{-9}$	$\simeq 1 \cdot 10^{-4}$	$\simeq 0.12$
e-MID 02	177	7252	$\simeq 6 \cdot 10^{-9}$	$\simeq 2 \cdot 10^{-13}$	$\simeq 0.7$	$\simeq 3 \cdot 10^{-8}$	$\simeq 5 \cdot 10^{-6}$	$\simeq 1$	$\simeq 8 \cdot 10^{-6}$	$\simeq 3 \cdot 10^{-1}$	$\simeq 0.08$
e-MID 03	179	6814	$\simeq 1 \cdot 10^{-12}$	$\simeq 2 \cdot 10^{-10}$	$\simeq 0.8$	$\simeq 3 \cdot 10^{-9}$	$\simeq 3 \cdot 10^{-3}$	$\simeq 0.7$	$\simeq 4 \cdot 10^{-15}$	$\simeq 8 \cdot 10^{-4}$	$\simeq 0.1$
e-MID 04	180	6136	$\simeq 9 \cdot 10^{-10}$	$\simeq 2 \cdot 10^{-7}$	$\simeq 0.9$	$\simeq 5 \cdot 10^{-15}$	$\simeq 8 \cdot 10^{-5}$	$\simeq 0.8$	$\simeq 5 \cdot 10^{-9}$	$\simeq 6 \cdot 10^{-4}$	$\simeq 0.13$
e-MID 05	176	6203	$\simeq 3 \cdot 10^{-10}$	$\simeq 5 \cdot 10^{-9}$	$\simeq 0.7$	$\simeq 2 \cdot 10^{-14}$	$\simeq 2 \cdot 10^{-3}$	$\simeq 0.7$	$\simeq 5 \cdot 10^{-9}$	$\simeq 1 \cdot 10^{-3}$	$\simeq 0.2$
e-MID 06	177	6132	$\simeq 1 \cdot 10^{-10}$	$\simeq 7 \cdot 10^{-12}$	$\simeq 0.7$	$\simeq 2 \cdot 10^{-13}$	$\simeq 3 \cdot 10^{-3}$	$\simeq 0.8$	$\simeq 8 \cdot 10^{-11}$	$\simeq 5 \cdot 10^{-1}$	$\simeq 0.14$
e-MID 07	178	6330	$\simeq 3 \cdot 10^{-6}$	$\simeq 3 \cdot 10^{-13}$	$\simeq 1$	$\simeq 1 \cdot 10^{-7}$	$\simeq 8 \cdot 10^{-6}$	$\simeq 1.2$	$\simeq 7 \cdot 10^{-12}$	$\simeq 7 \cdot 10^{-1}$	$\simeq 0.14$
e-MID 08	173	4767	$\simeq 3 \cdot 10^{-10}$	$\simeq 8 \cdot 10^{-13}$	$\simeq 0.8$	$\simeq 1 \cdot 10^{-9}$	$\simeq 1 \cdot 10^{-3}$	$\simeq 0.7$	$\simeq 3 \cdot 10^{-9}$	$\simeq 8 \cdot 10^{-1}$	$\simeq 0.07$
e-MID 09	156	2961	$\simeq 4 \cdot 10^{-11}$	$\simeq 3 \cdot 10^{-12}$	$\simeq 0.6$	$\simeq 1 \cdot 10^{-7}$	$\simeq 9 \cdot 10^{-5}$	$\simeq 0.7$	$\simeq 8 \cdot 10^{-10}$	$\simeq 2 \cdot 10^{-3}$	$\simeq 0.11$
e-MID 10	135	2743	$\simeq 2 \cdot 10^{-11}$	$\simeq 2 \cdot 10^{-9}$	$\simeq 0.7$	$\simeq 7 \cdot 10^{-13}$	$\simeq 5 \cdot 10^{-5}$	$\simeq 0.5$	$\simeq 5 \cdot 10^{-9}$	$\simeq 2 \cdot 10^{-1}$	$\simeq 0.05$

Table 5: Performance of Newton's, quasi-Newton and the fixed-point algorithm to solve the system of equations defining the (directed version of the) CReM, on a set of real-world WDNs. All algorithms stop because the condition $||\nabla \mathscr{L}(\vec{\theta})||_2 \leq 10^{-8}$ is satisfied. For what concerns accuracy, the two most accurate methods are Newton's and the quasi-Newton one; for what concerns speed, the fastest method is the fixed-point one. Only the results corresponding to the best choice of initial conditions are reported.

quasi-Newton method on the network size.

Lastly, while rescaling the system of equations defining the CReM improves neither the accuracy nor the speed of any of the three algorithms considered here, differences in their speed of convergence, caused by the choice of a particular set of initial conditions, are observable: the 'uniform' prescription outperforms the other ones (for both the undirected and the directed version of the CReM).

Sampling the CReM

As usual, let us comment on the algorithm to sample the CReM ensemble - for the sake of simplicity, in the undirected case. As for the UECM, this can be compactly achieved by implementing a two-step procedure, the only difference lying in the functional form of the distribution from which weights are sampled. Given a specific pair of vertices i, j (with $i < j$), the first link can be drawn from a Bernoulli distribution with probability p_{ij}^{UBCM} while the remaining $(w-1)$ ones can be drawn from an exponential distribution whose parameter reads $\theta_i + \theta_j$. The computational complexity of the sampling process is, again, $O(N^2)$ and the pseudo-code for explicitly sampling the CReM ensemble is summed up by Algorithm 4.

2.7 Discussion

The exercises carried out so far have highlighted a number of (stylized) facts concerning the performance of the three algorithms tested: in what follows, we will briefly sum them up.

Newton's method

Overall, Newton's method is very accurate - often, the most accurate one - in reproducing both the binary and the weighted constraints; moreover, it represent the only viable alternative when the most complicated models are considered (i.e. the UECM and the DECM, respectively defined by a system of $2N$ and $4N$ coupled, non-linear equations). However,

Algorithm 4 Sampling the CReM ensemble

```
1:  for $m = 1 \dots |E|$ do
2:      $\mathbf{W} = \mathbf{0}$;
3:      for $i = 1 \dots N$ do
4:          for $j = 1 \dots N$ and $j < i$ do
5:              if RandomUniform$[0, 1] \leq p_{i\alpha}^{\text{UBCM}}$ then
6:                  $w_{ij} = w_{ji} =$ RandomExponential$[\theta_i + \theta_j]$;
7:              else
8:                  $w_{ij} = w_{ji} = 0$;
9:              end if
10:         end for
11:     end for
12:     Ensemble$[m] = \mathbf{W}$;
13: end for
```

the time required to run Newton's method on a given model seems to be quite dependent on the network size, especially whenever the corresponding system of equations cannot be reduced - see the case of the undirected CReM, run on the Bitcoin Lightning Network. Since one of the reasons affecting the bad scaling of Newton's method with the network size is the evaluation of the Hessian matrix defining a given model, this algorithm has to be preferred for largely reducible networks.

Quasi-Newton method

For all the networks considered here, the quasi-Newton method we have implemented is nothing else than the diagonal version of the traditional Newton's method. Even if this choice greatly reduces the number of entries of the Hessian matrix which are needed (i.e. just N elements for the undirected version of the CReM, $2N$ elements for the UECM and the directed version of the CReM and $4N$ elements for the DECM) dimensionality may still represent an issue to achieve fast convergence. Moreover, since the diagonal approximation of the Hessian matrix is not necessarily always a good one, the quasi-Newton method may require more time than Newton's one to achieve the same level of accuracy in reproducing the constraints. However, when such an approximation is a good one, the 'regime' in which the quasi-Newton method outperforms the competitors seems to be the one of small, non-reducible networks (e.g. see the results concerning the DBCM run on the WTW) - althogh, in cases like these, Newton's method may still be a strong competitor.

Fixed-point method

From a purely theoretical point of view, the fixed-point recipe is the fastest one, since the time required to evaluate the generic n-th step is (only) due to the evaluation of the model-specific map at the $(n-1)$-th iteration. Strictly speaking, however, this holds true for a single step: if the number of steps required for convergence is large, in fact, the total amount of time required by the fixed-point method can be large as well. Overall, however, this algorithm has to be preferred for large, non-reducible networks: this is the case of the (undirected version of the) CReM, run on the 8-th snapshot of the Bitcoin Lightning Network (i.e. day 17-07-19) and requiring a bit more than one minute to achieve an accuracy of $\text{MRDE}_{\text{fixed-point}} \gtrsim 10^{-10}$ and of $\text{MRSE}_{\text{fixed-point}} \simeq 10^{-1}$; naturally, the method is not as accurate as Newton's one, for which $\text{MRDE}_{\text{Newton}} \gtrsim 10^{-12}$ and $\text{MRSE}_{\text{Newton}} \simeq 10^{-6}$ but is much faster as Newton's algorithm requires $\simeq 1.188$ seconds to converge.

Alternative techniques to improve accuracy and speed have been tested as well, as the one of coupling two of the algorithms considered above. In particular, we have tried to solve the (undirected version of the) CReM by running the fixed-point algorithm and using the solution of the latter as input for the quasi-Newton method. The results are reported in Table 6: as they clearly show, the coupled algorithm is indeed more accurate that the single methods composing it and much faster than the quasi-Newton one (for some snapshots, more accurate and even faster than Newton's method).

Techniques like these are, in general, useful to individuate better initial conditions than the completely random ones: a first run of the fastest method may be, in fact, useful to direct the most accurate algorithm towards the (best) solution. This is indeed the case, upon considering that

	Fixed-point + Quasi-Newton			
	MRDE	MADE	MRSE	Time (s)
BLN 24-01-18	$\simeq 2 \cdot 10^{-9}$	$\simeq 1.1 \cdot 10^{-7}$	$\simeq 1 \cdot 10^{-5}$	$\simeq 0.1$
BLN 25-02-18	$\simeq 1.3 \cdot 10^{-9}$	$\simeq 1.5 \cdot 10^{-6}$	$\simeq 1 \cdot 10^{-5}$	$\simeq 1.6$
BLN 30-03-18	$\simeq 1 \cdot 10^{-9}$	$\simeq 1.3 \cdot 10^{-7}$	$\simeq 7.8 \cdot 10^{-7}$	$\simeq 2.2$
BLN 13-07-18	$\simeq 7.5 \cdot 10^{-10}$	$\simeq 4.2 \cdot 10^{-4}$	$\simeq 5.3 \cdot 10^{-5}$	$\simeq 200$
BLN 19-12-18	$\simeq 7.5 \cdot 10^{-10}$	$\simeq 1.7 \cdot 10^{-8}$	$\simeq 1.1 \cdot 10^{-9}$	$\simeq 7$
BLN 30-01-19	$\simeq 6.2 \cdot 10^{-10}$	$\simeq 1.8 \cdot 10^{-5}$	$\simeq 4.1 \cdot 10^{-6}$	$\simeq 614$
BLN 01-03-19	$\simeq 5.7 \cdot 10^{-10}$	$\simeq 5.4 \cdot 10^{-6}$	$\simeq 9.1 \cdot 10^{-6}$	$\simeq 961$
BLN 17-07-19	$\simeq 4.9 \cdot 10^{-10}$	$\simeq 1.3 \cdot 10^{-3}$	$\simeq 3.5 \cdot 10^{-3}$	$\simeq 3350$

Table 6: Performance of the algorithm coupling fixed-point and quasi-Newton to solve the system of equations defining the (undirected version of the) CReM, on a set of real-world WUNs. The algorithm stops because the condition $||\nabla\mathscr{L}(\vec{\theta})||_2 \leq 10^{-8}$ is satisfied. As the results reveal, it is more accurate that the single methods composing it and much faster than the quasi-Newton one - for some snapshots, more accurate and even faster than Newton's method. Only the results corresponding to the best choice of initial conditions are reported.

the quasi-Newton method, now, stops because the condition $||\nabla\mathscr{L}(\vec{\theta})||_2 \leq 10^{-8}$ is satisfied - and not for having reached the limit of 10.000 steps.

We would like to end the discussion about the results presented in this contribution by explicitly mentioning a circumstance that is frequently met when studying economic and financial networks. When considering systems like these, the information about the number of neighbours of each node is typically not accessible: as a consequence, the models constraining both binary and weighted information cannot be employed as they have presented in this contribution.

Alternatives exist and rest upon the existence of some kind of relationship between binary and weighted constraints. In the case of undirected networks, such a relationship is usually written as

$$e^{-\theta_i} = \sqrt{z} s_i, \quad \forall\, i \tag{2.69}$$

and establishes that the Lagrange multipliers controlling for the degrees are linearly proportional to the strengths. If this is the case (or a valid reason exists for this to be the case), the expression for the probability that any two nodes are connected becomes

$$p_{ij}^{\text{dcGM}} = \frac{z s_i s_j}{1 + z s_i s_j}, \quad \forall\, i < j \tag{2.70}$$

the acronym standing for *degree-corrected Gravity Model* [63]. The (only) unknown parameter z must be numerically estimated by employing some kind of topological information; this is usually represented by (a proxy of) the network link density, used to instantiate the (only) likelihood condition

$$L(\mathbf{A}^*) = \langle L \rangle = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ (j<i)}}^{N} \frac{z s_i s_j}{1 + z s_i s_j}; \tag{2.71}$$

once the equation above has been solved, the set of coefficients $\{p_{ij}^{\text{dcGM}}\}_{i,j=1}^{N}$ can be either employed 1) to, first, estimate the degrees and, then, solve the UECM [64] or 2) within the CReM framework, via the identification $f_{ij} \equiv p_{ij}^{\text{dcGM}}$, to estimate the parameters controlling for the weighted constraints.

2.8 The 'NEMTROPY' package

As an additional result, we have released a comprehensive package, coded in Python, that implements the three aforementioned algorithms on all the ERGs considered in the present work (see also fig. 3). Its name is 'NEMTROPY' (an acronym standing for 'Network Entropy Maximization: a Toolbox Running On Python') and is available at [65].

With 'NEMTROPY', we aim at overcoming the limitations of currently available packages designed to optimize the likelihood function of the ERGs defined by local constraints[6].

[6]An early attempt is represented by the MAX&SAM module [66]: however, the chosen coding language (MATLAB) and its poor computational performances prevented network scientists to extensively use it.

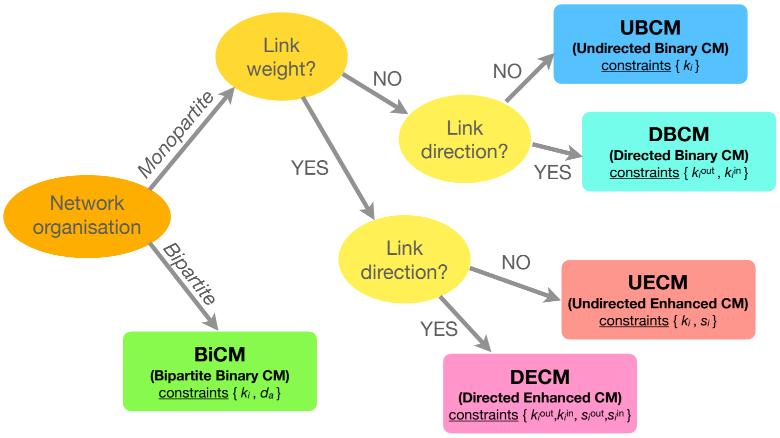


Figure 3: System diagram illustrating the models discussed in [31] and implemented in the 'NEMTROPY' package: it represents a sort of guide to individuate the best model for analysing the system at hand. Our package handles both monopartite and bipartite networks; while the latter ones have been considered only in their binary, undirected fashion, the former ones can be modeled either in a binary or a weighted fashion, allowing for both undirected and directed links.

The reasons behind the choice of writing 'NEMTROPY' in Python are many: 1) the possibility of leveraging on the programming expertise of scholars: in fact, some of the most used libraries for the analysis of complex networks are written in this language (e.g. `graph-tools`, `igraph`, `networkx`); 2) the possibility of using `numba`, a library translating Python functions into optimized machine code, hence guaranteeing performances (in terms of accuracy and scalability) comparable to those achievable by employing C or Fortran.

The 'NEMTROPY' module can be easily installed from the Python Package Index (PyPi) by tapping

```
pip install nemtropy
```

in your terminal[7]. In what follows, we will briefly introduce the basic types of the 'NEMTROPY' library and provide examples of codes illustrating how to solve ERGs and sample a random network from them.

2.8.1 Graph instances

The 'NEMTROPY' library deals with ERGs defined for a plethora of network types; indeed, different kinds of graph instances are accessible to the user, such as monopartite (undirected and directed) and bipartite (undirected). Moreover, monopartite graphs can be either binary or weighted.

Similarly to other Python modules for network analysis, a graph instance in 'NEMTROPY' can be initialized either using a graph edgelist or a graph adjacency matrix. Furthermore, when dealing with network reconstruction, the user can initialize a graph instance by using the degree sequence, the strength sequence or both.

As an example, the code to initialize the Zachary Karate Club in 'NEMTROPY', via its adjacency matrix, as an undirected graph instance reads as follows

```
import networkx as nx
from NEMtropy import UndirectedGraph
G = nx.karate_club_graph()
adj = nx.to_numpy_array(G)
g = UndirectedGraph(adjacency=adj_kar)
```

(notice that the `networkx` module is only used to retrieve the Zachary Karate Club adjacency matrix).

Let us now consider the case when only partial information about a graph is available, i.e. the degree sequence of an undirected network (say g). In this case, the first step is the initialization of an `UndirectedGraph` instance using the degree sequence:

[7]For more details about the supported Python versions and how to deal with potential compatibility errors, the reader is redirected to the Readme file that can be found on the Github page of the package. The choice of making our code available on Github was driven by the idea that anyone can collaborate to improve 'NEMTROPY' by adding new functions or fixing bugs.

```
from NEMtropy import UndirectedGraph
g = UndirectedGraph(degree_sequence=degree)
```

the second step, instead, is the resolution of the UBCM but we will discuss this in detail in the next paragraph.

The 'NEMTROPY' graph classes have also built-in functions returning some of the basic properties of a graph, such as

- `dseq` that returns the degree sequence of the graph;
- `strength_sequence` that returns the strength sequence of the graph;
- `n_nodes` that returns the number of nodes;
- `n_edges` that returns the number of edges.

Analogously, monopartite, directed and bipartite graph classes are characterized by their own built-in functions[8].

2.8.2 Binary graphs

Lets now discuss how the 'NEMTROPY' module can be used to solve ERGs for monopartite, binary graphs. In the undirected case, this can achieved by employing the following snippet of code

```
g = UndirectedGraph(adjacency=adj)
g.solve_tool(model="cm_exp",
             method="newton",
             initial_guess="random")
```

indicating that Newton's method is used to solve the UBCM by specifying a random initialization. For a directed graph, instead, the previous code becomes

```
g = DirectedGraph(adjacency=adj)
g.solve_tool(model="dcm_exp",
             method="newton",
             initial_guess="random")
```

[8]For more details, the reader is redirected to the Readme file that can be found on the Github page of the package.

where, again, we use Newton's method to solve the DBCM.

In general, the `solve_tool` function accepts the following arguments[9]:

- `model`: it specifies the ERG to be used;
- `method`: it specifies the optimization method to be used;
- `initial_guess`: it specifies the initial guess for the optimization process;
- `max_steps`: it specifies the maximum number of steps for the optimization process;
- `full_return`: if 'true', it returns more detailed information about the optimization.

2.8.3 Weighted graphs

Weighted ERGs can be either 'enhanced' or 'two-steps': since the arguments of `solve_tool` depend on the chosen model, we are going to describe them separately.

Enhanced methods

Enhanced ERGs preserve both the degree and the strength sequences and accept only integer weights. Given an undirected, weighted graph g, the following snippet of code solves the UECM

```
g = DirectedGraph(adjacency=adj_wei)
g.solve_tool(model="ecm_exp",
             method="newton",
             initial_guess="random")
```

analogously, the DECM can be solved by running the following code

[9]For the sake of simplicity, we are considering only part of the available arguments for `solve_tool`. For more details, the reader is redirected to the Readme file that can be found on the Github page of the package.

```
g = DirectedGraph(adjacency=adj_wei)
g.solve_tool(model="decm_exp",
             method="newton",
             initial_guess="random")
```

Two-step methods

As the name suggests, two-step models handle the binary and the weighted optimization steps in different moments. For example, a user can combine a deterministic, binary configuration with a weighted model and solve such a two-step model by running the following snippet of code

```
g = UndirectedGraph(strength_sequence=s_seq)
g.solve_tool(model="crema",
             method="newton",
             initial_guess="random",
             adjacency=adj_bin)
```

(where a graph instance is initialized using the strength sequence and the `crema` model is solved by employing a deterministic adjacency matrix for the binary part). Additionally to the parameters accepted by enhanced models, two-step models accept the following ones:

- `adjacency`: it can be a deterministic adjacency matrix or the probability matrix of a binary model (e.g. 'cm_exp'). If an adjacency matrix is provided, its entries must be either zero or one;
- `method_adjacency`: if the `adjacency` argument is a model, then it specifies the optimization method to be used;
- `initial_guess_adjacency`: if the `adjacency` argument is a model, then it specifies the initial guess to be used.

Two-step models can handle both integer and continuous weights.

2.8.4 Graph sampling

Once the `solve_tool` function has been used to compute the parameters determining the chosen ERG model, one or more configurations can be sampled by the ensemble induced by it using the `ensemble_sampler` function

```
g.ensemble_sampler(n=1, cpu_n=2, output_dir="sample/")
```

where `n` specifies the number of sampled graphs, `output_dir` specifies the path to the directory where graphs are saved and `cpu_n` specifies the number of CPUs used to compute these graphs.

Chapter 3

Detecting mesoscale structures by surprise

This chapter is devoted to present the results of the paper [42], accepted for publication on Communications Physics and dealing with our work about the generalization of the surprise-based formalism. Such an effort has led to the definition of a unified framework capable of detecting (modular and 'bimodular') mesoscale structures on networks (be they undirected, directed, binary, weighted) by assigning them a p-value. Specifically, we have considered six variants of the surprise: from a technical point of view, this amounts at employing six variants of the hypergeometric distribution. To illustrate the performance of our methods, we, first, test them on a variety of well-established, synthetic benchmarks and, then, run them on several real-world networks, including social, economic, financial and ecological ones.

3.1 Introduction

The importance of identifying the signature of some kind of mesoscopic organization in complex networks - be it due to the presence of communities or bipartite, core-periphery, bow-tie structures - can be hardly overestimated [67, 68], the best example of complex systems whose behavior is deeply affected by their mesoscopic structural organization being provided by financial networks [69, 70, 71, 72]. So far, much attention has been devoted to the detection of *binary mesoscale structures*, i.e. *communities* and, to a far less extent, *core-periphery structures*: the efforts to solve these problems have led to a number of approaches that are briefly sketched below (for a detailed review of them, see [73, 74]).

3.1.1 Three classes of methods

A universally accepted definition of community does not exist. As a matter of fact, the interpretation that we give to communities usually depends on the domain of application and on the particular study case. From an historical perspective, community detection has been initially approached by attempting a definition of 'communities' based on the concepts of *clustering coefficient*, *cliques* and *k-core* - used to quantify the internal cohesion of a group of nodes. Core-periphery structures, instead, have been defined in a purely top-down fashion, by imagining a fully connected subgraph (i.e. the core) surrounded by (peripherical) vertices exclusively linked to the first ones [69].

As stressed in [71], the deterministic character of these definitions makes their *tout court* application to real-world systems extremely difficult. This is the reason why the intuitive requirements that 'the number of internal edges is larger than the number of external edges' and that 'the core portion of a network is densely connected, while its periphery is loosely connected' [74] are, now, interpreted in a purely probabilistic way: a community has, thus, become a subgraph whose vertices have a larger probability to be interconnected than to be connected to any other vertex in the graph - and analogously for the core-periphery structure. In other words, the top-down approach defining a golden standard and looking for (deviations from) it has left the place to a bottom-up approach where structures are supposed to emerge as the result of non-trivial (i.e. non-casual) interactions between the nodes.

This change of perspective leads to a number of problems. The first one concerns the definition of models stating how edges are formed and has been solved by adopting the rich formalism defining the Exponential Random Graphs framework [43]; the second, and most important, one concerns the definition of a (statistically-sound) procedure for selecting the best model among the ones providing competing descriptions of the data. This has led to the identification of three broad classes of algorithms: according to our intuition, all of them implement some kind of *statistical inference*, the major difference lying in the way the corresponding test of hypothesis is implemented; from a practical point of view, instead, all these methods are designed for *optimization*, the functional form of the specific score function determining the class to which a given algorithm belongs[1].

The first class of methods

The most representative algorithms among those belonging to the first class are the ones based on *modularity* [75] whose definition reads

$$Q = \frac{1}{2L} \sum_{i \neq j} (a_{ij} - p_{ij}) \delta_{c_i, c_j} \tag{3.1}$$

where L is the number of edges, a_{ij} is the generic entry of the adjacency matrix, p_{ij} represents the probability of observing and edge between nodes i and j according to the chosen null-model and c_i is a label indicating the membership of node i. The traditional form of Q employs the sparse version of the Undirected Binary Configuration Model as a null model, i.e.

$$p_{ij} = \frac{k_i k_j}{2L}, \quad i < j \tag{3.2}$$

where k_i and k_j indicate the degrees of nodes i and j. By substituting p_{ij} into eq. (3.1) the most popular version of modularity is recovered, i.e.

$$Q = \frac{1}{2L} \sum_{i \neq j} \left(a_{ij} - \frac{k_i k_j}{2L} \right) \delta_{c_i, c_j}; \tag{3.3}$$

[1]Such a score function, indicating the 'quality' of a grouping of nodes, has to be optimized over the space of all possible partitions. Since the number of the latter ones, quantified by the N-th Bell number, is prohibitively large, one usually implements heuristic techniques.

different choices of the null-model lead to different versions of modularity (e.g. to account for bipartite structures [76], signed edges [77], correlations [78]).

Relying on the assumption that the presence of a modular organization is destroyed upon randomizing a network structure, modularity 'measures' how different an empirical graph is from its randomized version - a comparison that is supposed to reveal the presence of patterns characterizing the underlying network. While Q indeed embodies a comparison between the (empirical) adjacency matrix $\mathbf{A}$ and the matrix of probability coefficients $\mathbf{P}$ defining the benchmark, it does not provide any indication of the statistical significance of the recovered partition, the reason being that it is not designed as a proper statistical test (i.e. does not implement any proper test of hypothesis).

The algorithms prescribing to maximize a plain likelihood function belong to this first group as well. The rationale upon which these algorithms are based is that of fitting a generative network model on the data [79, 80, 81, 82, 83, 84]. The most popular example is provided by the Stochastic Block Model (SBM): given (the adjacency matrix of) a network, $\mathbf{A}$, and a partition, g, of its nodes into c communities, the likelihood that such a configuration is produced by the SBM reads

$$\mathscr{L}_{\text{SBM}}(\mathbf{A}|g) = \sum_{r,s} e_{rs} \ln\left(\frac{e_{rs}}{n_r n_s}\right) \tag{3.4}$$

where e_{rs} is the number of edges between groups r and s and n_r (n_s) is the number of nodes within group r (s).

Such a version of the SBM, however, does not account for the heterogeneity of the degrees, hence performing poorly in describing the community structure of many real-world networks. In order to overcome this limitation, Newman and Karrer introduced the degree-corrected Stochastic Block Model (dc-SBM) [82], 'producing' a likelihood

$$\mathscr{L}_{\text{dc-SBM}}(\mathbf{A}|g) = \sum_{r,s} e_{rs} \ln\left(\frac{e_{rs}}{e_r e_s}\right) \tag{3.5}$$

of describing the aforementioned configuration, with e_r (e_s) being the sum of the degrees of the nodes belonging to group r (s).

It is crucial to remark that eqs. (3.4) and (3.5) explicitly depend on the number of clusters c, an evidence implying that, in order to employ either the SBM or the dc-SBM, the number of modules partitioning the network must be known *a priori*: in fact, a straight maximization of the equations (3.4) and (3.5) over the entire set of possible partitions would output the trivial one where each vertex is a cluster on its own [82] - an issue known as *overfitting*.

The second class of methods

The aforementioned, major limitation is overcome by the algorithms belonging to the second class. They implement tests of hypothesis either *à la Fisher* or *à la Neyman-Pearson*, i.e. either defining a single benchmark (the null hypothesis of the first scenario) or two, alternative ones (the null and the alternative hypothesis of the second scenario): from a practical point of view, such a result is achieved by identifying the aforementioned benchmarks with proper probability distributions and the best partition of nodes with the one minimizing the corresponding p-value.

Surprise-based algorithms belong to this second group: as it has been shown in [38] - for a particular case; such a result will be generalized in what follows - optimizing (asymptotic) surprise amounts at carrying out a (sort of) Likelihood Ratio Test aimed at choosing between two alternative models.

The third class of methods

Hypothesis testing can be further refined by allowing for more than two hypotheses to be tested at a time: results of the kind are particularly useful for model selection and, in fact, have produced a plethora of criteria (e.g. the Akaike Information Criterion, the Bayesian Information Criterion and the Minimum Description Length) for singling out the best statistical model out of a basket of competing ones. Generally speaking, optimization, here, seeks for the maximum of a 'corrected' likelihood function embodying the trade-off between accuracy and parsimony of a description. An example of the algorithms belonging to this third class is represented by Infomap [36] searching for the number of clusters best compressing the data [85] - which is known to be a growing function of the number of blocks [86]. Other examples are represented by the recipes employing the SBM within a Bayesian framework (see [87] and the references therein).

3.2 Mesoscale structures detection via exact tests

With the work presented in this chapter, we pose ourselves within the second research line and adopt a bottom-up approach that prescribes to compare any empirical network structure with the outcome of a properly-defined benchmark model.

To this aim, we devise a unified framework for mesoscale structures detection based upon the score function called *surprise*[2], i.e. a p-value that can be assigned to any given partition of nodes, on both undirected and directed networks: while for binary community detection this is achieved by employing the *binomial hypergeometric distribution* [41, 37], with 'bimodular' structures like the bipartite and the core-periphery ones, one needs to consider its *multinomial* variant [38]. Here, however, we aim at making a step further, by extending the entire framework to the weighted case. As a result, we present a general, statistically-grounded approach to the problem of detecting mesoscale structures on networks via a unified, suprise-based framework[3].

[2]Our function is named *surprise* since it generalizes the function proposed in [41] for community detection. However, in our case it indicates a proper probability and not its logarithm, as in [41]. The same holds true for its asymptotic expression.

[3]The use of the hypergeometric distribution to carry out tests of hypothesis on networks is not novel: examples are provided by the papers [88] (where the authors introduce a method to provide a statistically-validated, monopartite projection of a bipartite network - the considered null hypothesis encoding the heterogeneity of the system), [89] (where the authors employ the same validation procedure to detect cores of communities within each set of nodes of a bipartite system) and [90] (where the authors extend the framework proposed in the aforementioned references to carry out a statistical validation of motifs observed in hypergraphs). For a review on the use of the hypergeometric distribution for network analyses see [91] and the references therein.

Surprise has recently received a lot of attention: the advantages of employing such a score function have been extensively discussed in [41, 38, 37, 92, 93, 94, 95] where researchers have tested and compared its performance from a purely numerical perspective. However, a characterization of the statistical properties of surprise is still missing: for this reason, we will, first, make an effort to 'translate' the problem of detecting a given mesoscale network structure into a proper *exact significance test*[4] and, then, show how the rich - yet, still underexplored - surprise-based formalism can properly answer such a question.

The basic equation underlying exact tests reads

$$\Pr(x \geq x^*) = \sum_{x \geq x^*} f(x) \tag{3.6}$$

and returns the probability of observing an outcome of the random variable X which is 'more extreme' than the realized one, i.e. x^*. In the setting above, f represents the distribution encoding the *null hypothesis* and $\Pr(x \geq x^*)$ - commonly known with the name of *p-value* - answers the question *is the realized value $X = x^*$ compatible with the (null) hypothesis that X is distributed according to f?*

The p-value constitutes the basic quantity for carrying out any significance test: hence, in what follows we will tackle the problem of detecting the signature of a *statistically-significant*, mesoscale network organization by individuating a specific test - i.e. a suitable functional form for f.

3.3 Detection of binary mesoscale structures

3.3.1 Modular structures detection

Within the surprise-based framework, the detection of binary, modular structures (a task usually referred to as 'binary community detection') is carried out via the identification

[4]To the best of our knowledge, the only other attempt of the kind - specifically, to detect communities - is the one in [96].

$$f(l_\bullet) \equiv \mathrm{H}(l_\bullet|V, V_\bullet, L) = \frac{\prod_{i=\bullet,\circ}\binom{V_i}{l_i}}{\binom{V}{L}} = \frac{\binom{V_\bullet}{l_\bullet}\binom{V_\circ}{l_\circ}}{\binom{V}{L}} = \frac{\binom{V_\bullet}{l_\bullet}\binom{V-V_\bullet}{L-l_\bullet}}{\binom{V}{L}} \tag{3.7}$$

i.e. by calculating the p-value

$$\mathscr{S} \equiv \sum_{l_\bullet \geq l_\bullet^*} f(l_\bullet) \tag{3.8}$$

of a *binomial hypergeometric distribution* whose parameters read as above. In this formalism, the $\bullet$ subscript will be meant to indicate quantities that are *internal* to communities while the $\circ$ subscript will be meant to indicate quantities that are *external* to communities. More precisely, the binomial coefficient $\binom{V_\bullet}{l_\bullet}$ enumerates the number of ways $l_\bullet$ links can be redistributed *within* communities, i.e. over the available $V_\bullet$ node pairs, while the binomial coefficient $\binom{V_\circ}{l_\circ}$ enumerates the number of ways the remaining $l_\circ = L - l_\bullet$ links can be redistributed *between* communities, i.e. over the remaining $V_\circ = V - V_\bullet$ node pairs. Notice that, although $l_\bullet$ is 'naturally' bounded by the value $V_\bullet$, it cannot exceed L - whence the usual requirement $l_\bullet \in [l_\bullet^*, \min\{L, V_\bullet\}]$.

From a merely statistical point of view, surprise 'considers' a network as a population of V node pairs, L of which have been drawn; out of the L extracted ones, $l_\bullet$ node pairs have the desired feature of being 'internal' to communities, since they connect some of the node pairs belonging to the set $V_\bullet$. Hence, for a given partition of nodes into communities, $\mathscr{S}$ quantifies the probability of observing at least $l_\bullet^*$ 'successes' (i.e. intra-cluster edges) out of L draws: the lower this probability, the 'more surprising' the observation of the corresponding partition, hence the 'better' the partition itself.

Asymptotic results

We can gain more insight into the surprise-based formalism above upon deriving an asymptotic expression for $\mathscr{S}$ [97]. To this aim, let us consider that it can be simplified upon Stirling-approximating the binomial coefficients that appear within it. By exploiting the recipe $n! \simeq \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$, $\mathscr{S}$ can be rewritten as

$$\mathscr{S} \simeq \sum_{l_\bullet \geq l_\bullet^*} A(l_\bullet)\left[\frac{\mathrm{Ber}(V, L, p)}{\prod_{i=\bullet,\circ} \mathrm{Ber}(V_i, l_i, p_i)}\right] \tag{3.9}$$

where the expression

$$\mathrm{Ber}(x, y, z) = z^y(1-z)^{x-y} \tag{3.10}$$

defines a *Bernoulli* probability mass function, the parameters appearing in eq. (3.9) read $p = \frac{L}{V}$ and $p_i = \frac{l_i}{V_i}$ and the coefficient in front of the sum is

$$A(l_\bullet) = \sqrt{\frac{\sigma^2}{2\pi \prod_{i=\bullet,\circ} \sigma_i^2}} \tag{3.11}$$

with $\sigma^2 = Vp(1-p)$ and $\sigma_i^2 = V_i p_i (1-p_i)$. Equation (3.9) makes it explicit that employing $\mathscr{S}$ for binary community detection ultimately amounts at comparing the description of a networked configuration provided by the Random Graph Model (RGM), and encoded into the expression

$$\mathrm{Ber}(V, L, p) = p^L(1-p)^{V-L} \tag{3.12}$$

with the description of the same configuration provided by the Stochastic Block Model (SBM) [82], and encoded into the expression

$$\prod_{i=\bullet,\circ} \mathrm{Ber}(V_i, l_i, p_i) = p_\bullet^{l_\bullet}(1-p_\bullet)^{V_\bullet - l_\bullet} \cdot p_\circ^{l_\circ}(1-p_\circ)^{V_\circ - l_\circ} \tag{3.13}$$

where $p_\bullet = \frac{l_\bullet}{V_\bullet}$ and $p_\circ = \frac{l_\circ}{V_\circ}$. Naturally, the SBM takes as input the two sets indexed by $\bullet$ and $\circ$ and distinguish the connections found 'within the clusters' - contributing to the probability of the whole configuration with the term $\mathrm{Ber}(V_\bullet, l_\bullet, p_\bullet) = p_\bullet^{l_\bullet}(1-p_\bullet)^{V_\bullet - l_\bullet}$ - from the ones found 'between the clusters' - contributing to the probability of the whole configuration with the term $\mathrm{Ber}(V_\circ, l_\circ, p_\circ) = p_\circ^{l_\circ}(1-p_\circ)^{V_\circ - l_\circ}$.

Notice also that the asymptotic expression of the surprise guarantees that the parameters of the null models defining it are tuned according to the maximum-of-the-likelihood principle. To see this explicitly, let us consider $\text{Ber}(x, y, z)$ whose log-likelihood reads

$$\mathscr{L} = y \ln z + (x - y) \ln(1 - z); \tag{3.14}$$

upon maximizing it with respect to z, one finds $z = \frac{y}{x}$.

The way $\mathscr{S}$ works, i.e. by comparing two different null models, is reminiscent of more traditional likelihood ratio tests, where a null hypothesis H_0 (in our case, a given partition is compatible with the RGM) is tested against an alternative hypothesis H_1 (in our case, the given partition is compatible with the SBM): as the asymptotic expression of the surprise clarifies, minimizing it amounts at finding the partition least likely to occour under the RGM than under the SBM.

3.3.2 Bimodular structures detection

The surprise-based framework can be easily extended to detect what can be called 'bimodular structures', a term that will be used to compactly indicate core-periphery [98, 99, 100] and bipartite structures [101, 102]. The reason for adopting such a terminology lies in the evidence that both kinds of structures are defined by bimodular partitions, i.e. partitions of nodes into two different groups.

As shown elsewhere [38], the issue of detecting binary, 'bimodular' structures can be addressed by considering a *multivariate* (or *multinomial*) *hypergeometric distribution*, i.e. by identifying

$$\begin{aligned} f(l_\bullet, l_\circ) &\equiv \text{MH}(l_\bullet, l_\circ | V, V_\bullet, V_\circ, L) \\ &= \frac{\prod_{i=\bullet,\circ,\top} \binom{V_i}{l_i}}{\binom{V}{L}} = \frac{\binom{V_\bullet}{l_\bullet}\binom{V_\circ}{l_\circ}\binom{V_\top}{l_\top}}{\binom{V}{L}} = \frac{\binom{V_\bullet}{l_\bullet}\binom{V_\circ}{l_\circ}\binom{V-(V_\bullet+V_\circ)}{L-(l_\bullet+l_\circ)}}{\binom{V}{L}} \end{aligned} \tag{3.15}$$

where $V_\top \equiv V - (V_\bullet + V_\circ)$ indicates the number of node pairs between the modules $\bullet$ and $\circ$ and $l_\top \equiv L - (l_\bullet + l_\circ)$ indicates the number of links that must be assigned therein. While the binomial coefficient $\binom{V_\bullet}{l_\bullet}$ enumerates the number of ways $l_\bullet$ links can redistributed *within* the first module (e.g. the core portion) and the binomial coefficient $\binom{V_\circ}{l_\circ}$ enumerates the number of ways $l_\circ$ links can redistributed *within* the second module (e.g. the periphery portion), the third binomial coefficient $\binom{V-(V_\bullet+V_\circ)}{L-(l_\bullet+l_\circ)}$ enumerates the number of ways the remaining $L-(l_\bullet+l_\circ)$ links can be redistributed *between* the first and the second module, i.e. over the remaining $V-(V_\bullet+V_\circ)$ node pairs. This choice induces the definition of the *binary bimodular surprise*

$$\mathscr{S}_{/\!/} \equiv \sum_{l_\bullet \geq l_\bullet^*} \sum_{l_\circ \geq l_\circ^*} f(l_\bullet, l_\circ); \tag{3.16}$$

analogously to the univariate case, $l_\bullet$ and $l_\circ$ are 'naturally' bounded by the values $V_\bullet$ and $V_\circ$ - notice, however, that the sum $l_\bullet + l_\circ$ cannot exceed L (although it may not reach such a value, e.g. in case $V_\bullet + V_\circ < L$).

Asymptotic results

Analogously to the univariate case, the asymptotic expression for $\mathscr{S}_{/\!/}$ can be derived upon Stirling-approximating the binomial coefficients appearing within it. This time, the recipe $n! \simeq \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$ leads to

$$\mathscr{S}_{/\!/} \simeq \sum_{l_\bullet \geq l_\bullet^*} \sum_{l_\circ \geq l_\circ^*} B(l_\bullet, l_\circ) \left[\frac{\text{Ber}(V, L, p)}{\prod_{i=\bullet,\circ,\top} \text{Ber}(V_i, l_i, p_i)}\right] \tag{3.17}$$

where, as before, $\text{Ber}(x, y, z) = z^y(1-z)^{x-y}$ defines a Bernoulli probability mass function and the parameters read $p = \frac{L}{V}$ and $p_i = \frac{l_i}{V_i}$; the numerical coefficient appearing in front of the whole expression, now, reads

$$B(l_\bullet, l_\circ) = \frac{1}{2\pi}\sqrt{\frac{\sigma^2}{\prod_{i=\bullet,\circ,\top} \sigma_i^2}} \tag{3.18}$$

with $\sigma^2 = Vp(1-p)$ and $\sigma_i^2 = V_i p_i(1-p_i)$. The quantity $\mathscr{S}_{/\!/}$ compares the description of a networked configuration provided by the RGM, and encoded into the expression $\text{Ber}(V, L, p) = p^L(1-p)^{V-L}$, with the description of the same configuration provided by the SBM (now, defined by three - instead of two - different blocks), 'represented' by the denominator of the expression defined in eq. (3.17), i.e.

$$\prod_{i=\bullet,\circ,\top} \text{Ber}(V_i, l_i, p_i) = p_\bullet^{l_\bullet}(1-p_\bullet)^{V_\bullet - l_\bullet} \cdot p_\circ^{l_\circ}(1-p_\circ)^{V_\circ - l_\circ} \cdot p_\top^{l_\top}(1-p_\top)^{V_\top - l_\top}. \tag{3.19}$$

3.4 Detection of weighted mesoscale structures

3.4.1 Modular structures detection

Within the surprise-based framework, the problem of detecting binary communities has been rephrased as an aleatory experiment whose random variable is the number of links within communities. Interestingly enough, such an experiment can be easily mapped into a counting problem, allowing us to interpret $\mathscr{S}$ as indicating the number of configurations whose number of 'internal' links (i.e. within communities) is larger than the observed one.

When dealing with weighted networks, we would like to proceed along similar guidelines and consider the total, 'internal' weight as our new random variable, to be redistributed across the available node pairs. Adopting this approach has three major consequences: 1) weights must be considered as composed by an integer number of binary links, 2) each node pair must be allowed to be occupied by more than one link and 3) the total weight must be allowed to vary even beyond the network size (when handling real-world networks, the case $W \gg V$ is often encountered).

In this case, the proper setting to define an aleatory experiment satisfying the requests above is provided by the so-called *stars and bars* model, a combinatorial technique that has been introduced to handle the counting of configurations with multiple occupancies. Basically, the problem of counting in how many ways $w_\bullet$ particles (our links) can be redistributed among $V_\bullet$ boxes (our node pairs), while allowing more than one particle to occupy each box, can be tackled by allowing *both* the particles *and* the bars 'delimiting' the boxes to be permuted [103]. Since $V_\bullet$ boxes are delimited by $V_\bullet - 1$ bars, a term like $\binom{V_\bullet + w_\bullet - 1}{w_\bullet}$ is needed.

In order to better grasp the meaning of such a term, let us make a simple example. Let us imagine to observe a network with three nodes and two links, carrying a weight of 1 and 2, respectively. Now, were we interested in a purely binary analysis, we may ask ourselves in how many ways we could place the two links among the $\frac{N(N-1)}{2} = \frac{3(3-1)}{2} = 3$ available pairs: the answer is provided by the 'binary' binomial coefficient $\binom{V_\bullet}{l_\bullet} = \binom{3}{2} = 3$. The implicit assumption we make is that the three links must not occupy the same node pairs - otherwise the total number of connections wouldn't be preserved.

This perspective changes from the purely weighted point of view. Since we are now interested in preserving just the total weight of our network, irrespectively of the number of connections it is placed upon, the number of admissible configurations amounts precisely at $\binom{V_\bullet + w_\bullet - 1}{w_\bullet} = \binom{2+3}{3} = 10$. Such a number is larger than before since, now, weights are 'disaggregated' into binary links and multiple occupations of the latter ones are allowed - see also fig. 4.

The considerations above lead us to generalize the community detection problem to the weighted case by identifying

$$\begin{aligned} f(w_\bullet) &\equiv \mathrm{NH}(w_\bullet | V + W, W, V_\bullet) \\ &= \frac{\prod_{i=\bullet,\circ} \binom{V_i + w_i - 1}{w_i}}{\binom{V+W-1}{W}} = \frac{\binom{V_\bullet + w_\bullet - 1}{w_\bullet}\binom{V_\circ + w_\circ - 1}{w_\circ}}{\binom{V+W-1}{W}} \\ &= \frac{\binom{V_\bullet + w_\bullet - 1}{w_\bullet}\binom{(V-V_\bullet)+(W-w_\bullet)-1}{W-w_\bullet}}{\binom{V+W-1}{W}} \end{aligned} \qquad (3.20)$$

i.e. by replacing the binomial hypergeometric distribution considered in the purely binary case with a *negative hypergeometric distribution*, a choice inducing the definition of the *weighted surprise*

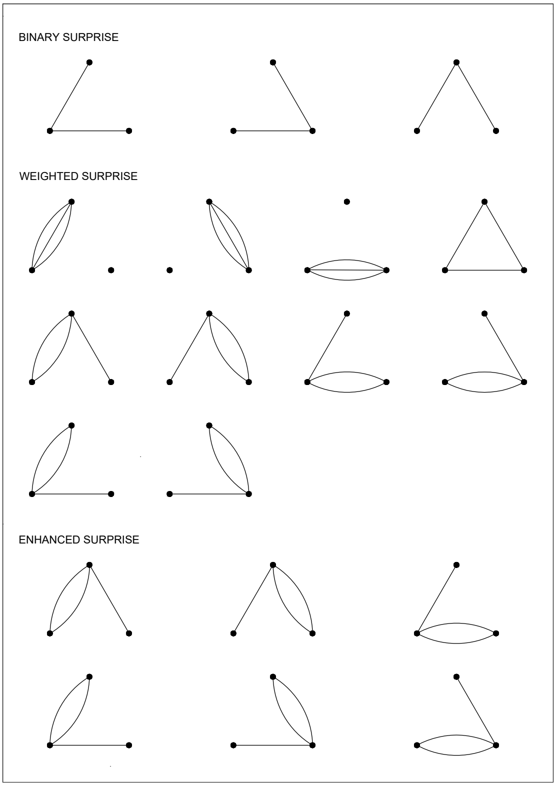


Figure 4: Graphical comparison of the three different ways of counting the admissible configurations when dealing with the purely binary, the purely weighted and the enhanced surprise. Let us imagine to observe a network with three nodes and two links, carrying a weight of 1 and 2, respectively. Were we interested in a purely binary analysis, we may ask ourselves in how many ways we could place the two links among the 3 available pairs: the answer is provided by the 'binary' binomial coefficient $\binom{V_\bullet}{l_\bullet} = \binom{3}{2} = 3$. Were we interested in a purely weighted analysis, we may ask ourselves in how many ways we could place the two links among the 3 available pairs while preserving the total weight of our network, irrespectively of the number of connections it is placed upon; the number of admissible configurations becomes $\binom{V_\bullet + w_\bullet - 1}{w_\bullet} = \binom{2+3}{3} = 10$. Such a number is larger than before since, now, weights are 'disaggregated' into binary links and multiple occupations of the latter ones are allowed.

$$\mathscr{W} \equiv \sum_{w_\bullet \geq w_\bullet^*} f(w_\bullet) \tag{3.21}$$

where the binomial coefficient $\binom{V_\bullet + w_\bullet - 1}{w_\bullet} = \binom{V_\bullet + w_\bullet - 1}{V_\bullet - 1}$ enumerates the number of ways $w_\bullet$ links can be redistributed *within* communities, i.e. over the available $V_\bullet$ node pairs, and the binomial coefficient $\binom{V_\circ + w_\circ - 1}{w_\circ} = \binom{V_\circ + w_\circ - 1}{V_\circ - 1}$ enumerates the number of ways the remaining $w_\circ = W - w_\bullet$ links can be redistributed *between* communities, i.e. over the remaining $V_\circ = V - V_\bullet$ node pairs. Differently from the binary case, the sum ranges up to the maximum empirical weight of the network, i.e. $w_\bullet \in [w_\bullet^*, W]$.

Asymptotic results

The asymptotic expression for $\mathscr{W}$ can be deduced by following the same reasoning that has allowed us to derive the asymptotic expression for $\mathscr{S}$. Stirling-approximating the binomial coefficients entering into the definition of $\mathscr{W}$ leads to the writing

$$\mathscr{W} \simeq \sum_{w_\bullet \geq w_\bullet^*} C(w_\bullet) \left[\frac{\text{Geo}(V, W, q)}{\prod_{i=\bullet,\circ} \text{Geo}(V_i, w_i, q_i)} \right] \tag{3.22}$$

where the expression

$$\text{Geo}(x, y, z) = z^y (1 - z)^x \tag{3.23}$$

defines a *geometric* probability mass function and the parameters appearing in eq. (3.22) read $q = \frac{W}{V+W-1}$ and $q_i = \frac{w_i}{V_i + w_i - 1}$. In the weighted case, the Bernoulli probability mass function appearing in the asymptotic expression of $\mathscr{S}$ is replaced by a geometric probability mass function: this implies that (asymptotically) the comparison is, now, carried out between the description of a networked configuration provided by the Weighted Random Graph Model (WRGM), and encoded into the expression

$$\text{Geo}(V, W, q) = q^W (1 - q)^V \tag{3.24}$$

with the description of the same configuration provided by the Weighted Stochastic Block Model (WSBM) and encoded into the expression

$$\prod_{i=\bullet,\circ} \text{Geo}(V_i, w_i, q_i) = q_\bullet^{w_\bullet}(1-q_\bullet)^{V_\bullet} \cdot q_\circ^{w_\circ}(1-q_\circ)^{V_\circ} \tag{3.25}$$

where $q_\bullet = \frac{w_\bullet}{V_\bullet + w_\bullet}$ and $q_\circ = \frac{w_\circ}{V_\circ + w_\circ}$. As for the binary case, the asymptotic expression of the weighted surprise clarifies that minimizing it amounts at finding the partition least likely to occour under the WRGM than under the WSBM.

As in the binary case, the parameters characterizing the geometric probability mass functions defining the asymptotic weighted surprise can be estimated via the maximum-of-the-likelihood principle; according to it, the log-likelihood of the expression Geo(x, y, z) reads

$$\mathscr{L} = y \ln z + x \ln(1-z) \tag{3.26}$$

and maximizing it with respect to z leads to the result $z = \frac{y}{x+y}$: hence, one can pose $q \simeq \frac{W}{V+W}$ and $q_i \simeq \frac{w_i}{V_i + w_i}$.

The numerical coefficient appearing in front of the whole expression, instead, reads

$$C(w_\bullet) = \sqrt{\frac{\mu}{2\pi \prod_{i=\bullet,\circ} \mu_i}} \tag{3.27}$$

with $\mu \simeq Vq$ and $\mu_i \simeq V_i q_i$.

3.4.2 Bimodular structures detection

Let us now introduce the third generalization of the suprise-based formalism: following the same line of reasoning that led us to approach the detection of binary 'bimodular' structures by considering the multinomial analogue of the distribution introduced for binary community detection, we focus on the *multinomial* (or *multivariate*) *negative hypergeometric distribution*, i.e.

$$\begin{aligned} f(w_\bullet, w_\circ) &\equiv \text{MNH}(w_\bullet, w_\circ | V+W, W, V_\bullet, V_\circ) \\ &= \frac{\prod_{i=\bullet,\circ,\top} \binom{V_i+w_i-1}{w_i}}{\binom{V+W-1}{W}} = \frac{\binom{V_\bullet+w_\bullet-1}{w_\bullet}\binom{V_\circ+w_\circ-1}{w_\circ}\binom{V_\top+w_\top-1}{w_\top}}{\binom{V+W-1}{W}} \\ &= \frac{\binom{V_\bullet+w_\bullet-1}{w_\bullet}\binom{V_\circ+w_\circ-1}{w_\circ}\binom{V-(V_\bullet+V_\circ)+W-(w_\bullet+w_\circ)-1}{W-(w_\bullet+w_\circ)}}{\binom{V+W-1}{W}}; \end{aligned} \tag{3.28}$$

while the binomial coefficient $\binom{V_\bullet+w_\bullet-1}{w_\bullet}$ enumerates the number of ways $w_\bullet$ links can redistributed *within* the first module (e.g. the core portion), the binomial coefficient $\binom{V_\circ+w_\circ-1}{w_\circ}$ enumerates the number of ways $w_\circ$ links can be redistributed *within* the second module (e.g. the periphery portion) and the binomial coefficient $\binom{V-(V_\bullet+V_\circ)+W-(w_\bullet+w_\circ)-1}{W-(w_\bullet+w_\circ)}$ enumerates the number of ways the remaining $w_\top \equiv W-(w_\bullet+w_\circ)$ links can be redistributed *between* the first and the second module, i.e. over the remaining $V_\top \equiv V-(V_\bullet+V_\circ)$ node pairs. Such a position induces the definition of the *weighted bimodular surprise*

$$\mathscr{W}_{/\!/} \equiv \sum_{w_\bullet \geq w_\bullet^*} \sum_{w_\circ \geq w_\circ^*} f(w_\bullet, w_\circ); \tag{3.29}$$

as for the (weighted) community detection, weights are understood as integer numbers - equivalently, as composed by an integer number of binary links. For what concerns the limits of the summations, $w_\bullet$ and $w_\circ$ are 'naturally' bounded by W; notice, however, that the sum $w_\bullet + w_\circ$ itself cannot exceed such a value.

Asymptotic results

Let us now derive the asymptotic expression for $\mathscr{W}_{/\!/}$. As usual, let us Stirling-approximate the binomial coefficients entering into the definition of $\mathscr{W}_{/\!/}$; such a simplification leads us to the expression

$$\mathscr{W}_{/\!/} \simeq \sum_{w_\bullet \geq w_\bullet^*} \sum_{w_\circ \geq w_\circ^*} D(w_\bullet, w_\circ) \left[\frac{\text{Geo}(V, W, q)}{\prod_{i=\bullet,\circ,\top} \text{Geo}(V_i, w_i, q_i)}\right] \tag{3.30}$$

where, as before, $\text{Geo}(x,y,z) = z^y(1-z)^x$ defines a geometric probability mass function and the parameters read $q = \frac{W}{V+W-1}$ and $q_i = \frac{w_i}{V_i+w_i-1}$ but can be approximated as $q \simeq \frac{W}{V+W}$ and $q_i \simeq \frac{w_i}{V_i+w_i}$, according to the maximum-of-the-likelihood principle; the numerical coefficient multiplying the whole expression reads

$$D(w_\bullet, w_\circ) = \frac{1}{2\pi}\sqrt{\frac{\mu}{\prod_{i=\bullet,\circ,\top}\mu_i}} \tag{3.31}$$

with $\mu \simeq Vq$ and $\mu_i \simeq V_i q_i$. The analysis of $\mathscr{W}_{/\!/}$ in the asymptotic regime reveals that it compares the description of a networked configuration provided by the WRGM, and encoded into the expression $\text{Geo}(V,W,q) = q^W(1-q)^V$, with the description of the same configuration provided by the WSBM (now, defined by three - instead of two - different blocks), 'represented' by the denominator of the expression defined in eq. (3.30), i.e.

$$\prod_{i=\bullet,\circ,\top}\text{Geo}(V_i,w_i,q_i) = q_\bullet^{w_\bullet}(1-q_\bullet)^{V_\bullet}\cdot q_\circ^{w_\circ}(1-q_\circ)^{V_\circ}\cdot q_\top^{w_\top}(1-q_\top)^{V_\top}. \tag{3.32}$$

3.5 Enhanced detection of mesoscale structures

3.5.1 Modular structures detection

The recipe to detect communities on weighted networks can be further refined to account for the information encoded into the total number of links, beside the one provided by the total weight. Generally speaking, this can be realized by 'combining' two of the distributions introduced above. To this aim, let us proceed in a two-step fashion: first, let us recall that the number of ways L links can be placed among V node pairs, in such a way that $l_\bullet$ connections are 'internal' to the clusters while the remaining $L - l_\bullet$ ones are, instead, 'external' is precisely

$$\text{H}(l_\bullet|V,V_\bullet,L) = \frac{\binom{V_\bullet}{l_\bullet}\binom{V-V_\bullet}{L-l_\bullet}}{\binom{V}{L}}; \tag{3.33}$$

now, for each of the binary configurations listed above, $W - L$ links remain to be assigned: while $w_\bullet - l_\bullet$ of them must be placed within the clusters, on top of the $l_\bullet$ available 'internal' links, the remaining $(W - L) - (w_\bullet - l_\bullet)$ ones must be placed between the clusters, on top of the $L - l_\bullet$ available, inter-cluster connections. Hence, the 'conditional' negative hypergeometric distribution reading

$$\mathrm{NH}(w_\bullet|W, W-L, l_\bullet) = \frac{\binom{l_\bullet+(w_\bullet-l_\bullet)-1}{w_\bullet-l_\bullet}\binom{(L-l_\bullet)+(W-L)-(w_\bullet-l_\bullet)-1}{(W-L)-(w_\bullet-l_\bullet)}}{\binom{L+(W-L)-1}{W-L}} \tag{3.34}$$

remains naturally defined; now, 'combining' the two distributions above, simplifying and re-arranging, the generic term of the *enhanced hypergeometric distribution* can be rewritten as

$$\begin{aligned}\mathrm{EH}(l_\bullet, w_\bullet|V, V_\bullet, L, W) &= \mathrm{H}(l_\bullet|V, V_\bullet, L) \cdot \mathrm{NH}(w_\bullet|W, W-L, l_\bullet)\\ &= \frac{\binom{V_\bullet}{l_\bullet}\binom{V_\circ}{l_\circ}}{\binom{V}{L}} \cdot \frac{\binom{w_\bullet-1}{w_\bullet-l_\bullet}\binom{w_\circ-1}{w_\circ-l_\circ}}{\binom{W-1}{W-L}}\end{aligned} \tag{3.35}$$

with a clear meaning of the symbols. An analytical characterization of it is provided into Appendix B.3: for the moment, let us simply notice that the definition provided above works for the values $0 < l_\bullet < L$. By posing $f(l_\bullet, w_\bullet) \equiv \mathrm{EH}(l_\bullet, w_\bullet|V, V_\bullet, L, W)$, our novel distribution induces the definition of the *enhanced surprise*, i.e.

$$\mathscr{E} \equiv \sum_{l_\bullet \geq l_\bullet^*} \sum_{w_\bullet \geq w_\bullet^*} f(l_\bullet, w_\bullet); \tag{3.36}$$

although $l_\bullet$ and $w_\bullet - l_\bullet$ are 'naturally' bounded by $V_\bullet$ and $W - L$, respectively, the former one cannot exceed L.

In order to better understand how the enhanced surprise works, let us consider again the aforementioned example: given a network with three nodes and two links, carrying a weight of 1 and 2, respectively, we observe $\binom{V_\bullet}{l_\bullet} = \binom{3}{2} = 3$ (purely binary) configurations with exactly the same number of links and $\binom{V_\bullet+w_\bullet-1}{w_\bullet} = \binom{2+3}{3} = 10$ (purely weighted) configurations with exactly the same total weight. If we, now, constrain both the total number of links and the total weight of the network, the number of admissible configurations becomes $\binom{V_\bullet}{l_\bullet}\binom{w_\bullet-1}{w_\bullet-l_\bullet}=\binom{3}{2}\binom{3-1}{3-2}=3\cdot2=6$, as it can be easily verified upon explicitly listing them. Naturally, the configurations 'admissible' by the enhanced surprise are a subset of the configurations 'admissible' by the weighted surprise, i.e. precisely the ones with the desired number of links (see also fig. 4).

Asymptotic results

Analogously to the other functionals, the asymptotic expression of $\mathscr{E}$ can be derived by Stirling-approximating the binomial coefficients entering into its definition as well:

$$\mathscr{E}\simeq\sum_{l_\bullet\geq l_\bullet^*}\sum_{w_\bullet\geq w_\bullet^*}E(l_\bullet,w_\bullet)\left[\frac{\text{BF}(V,L,W,p,r)}{\prod_{i=\bullet,\circ}\text{BF}(V_i,l_i,w_i,p_i,r_i)}\right]; \tag{3.37}$$

in the formula above, the expression

$$\text{BF}(x,y,u,z,t)=z^y(1-z)^{x-y}\cdot t^{u-y}(1-t)^y=\text{Ber}(x,y,z)\cdot\text{Geo}(y,u-y,t) \tag{3.38}$$

defines a *Bose-Fermi* probability mass function [58]. While a Bernoulli probability mass function characterizes the asymptotic behavior of the purely binary surprise and a geometric probability mass function characterizes the asymptotic behavior of the purely weighted surprise, the enhanced surprise is asymptotically characterized by a distribution whose functional form is halfway between the two previous ones: while its Bernoulli-like portion controls for the 'presence' of the links, its 'conditional' geometric-like portion controls for the magnitude of the 'remaining' weights. To stress its 'mixed' character, such a distribution has been named 'Bose-Fermi': remarkably, it can be retrieved within the Exponential Random Graphs framework as a consequence of Shannon entropy maximization, constrained to simultaneously reproduce both the total number of links and the total weight of a network [58, 104].

The parameters appearing in eq. (3.37) read $p = \frac{L}{V}$, $p_i = \frac{l_i}{V_i}$ and $r = \frac{W-L}{W-1}$, $r_i = \frac{w_i - l_i}{w_i - 1}$; while the first class of parameters can be tuned according to the maximum-of-the-likelihood principle, the ones belonging to the second class can be approximated according to the same recipe. To see this explicitly, let us consider BF(x, y, u, z, t), whose log-likelihood reads

$$\mathscr{L} = y \ln z + (x - y) \ln(1 - z) + (u - y) \ln t + y \ln(1 - t); \tag{3.39}$$

upon maximizing it with respect to z, one finds $z = \frac{y}{x}$; upon maximizing it with respect to t one finds $t = \frac{u-y}{u}$ - hence, one can pose $r \simeq \frac{W-L}{W}$, $r_i \simeq \frac{w_i - l_i}{w_i}$.

The numerical coefficient multiplying the whole expression is defined as

$$E(l_\bullet, w_\bullet) = \frac{1}{2\pi} \sqrt{\frac{\sigma^2 \mu}{\prod_{i=\bullet,\circ} \sigma_i^2 \mu_i}} \tag{3.40}$$

where $\sigma^2 = Vp(1-p)$, $\sigma_i^2 = V_i p_i (1 - p_i)$, $\mu \simeq Lr$ and $\mu_i \simeq l_i r_i$.

Similarly to the other cases, the asymptotic expression of the enhanced suprise compares the description of a networked configuration provided by the Enhanced Random Graph Model (ERGM), and encoded into the expression BF$(V, L, W, p, r) = p^L (1-p)^{V-L} \cdot r^{W-L} (1-r)^L$, with the description of the same configuration provided by its block-wise counterpart, i.e. the Enhanced Stochastic Block Model (ESBM), encoded into the expression $\prod_{i=\bullet,\circ}$ BF$(V_i, l_i, w_i, p_i, r_i)$.

3.5.2 Bimodular structures detection

The last generalization of surprise concerns its use for the detection of 'bimodular' structures within the enhanced framework. This amounts at considering the following 'multinomial variant' of the enhanced hypergeometric distribution

$$\text{MEH}(l_\bullet, l_\circ, w_\bullet, w_\circ | V, V_\bullet, V_\circ, L, W) = \\ = \frac{\binom{V_\bullet}{l_\bullet}\binom{V_\circ}{l_\circ}\binom{V_\top}{l_\top}}{\binom{V}{L}} \cdot \frac{\binom{w_\bullet - 1}{w_\bullet - l_\bullet}\binom{w_\circ - 1}{w_\circ - l_\circ}\binom{w_\top - 1}{w_\top - l_\top}}{\binom{W-1}{W-L}} \tag{3.41}$$

where $V_\top \equiv V - (V_\bullet + V_\circ)$ indicates the number of node pairs between the modules $\bullet$ and $\circ$ and $l_\top \equiv L - (l_\bullet + l_\circ)$ indicates the number of links that must be assigned therein. Analogously, $w_\top = W - (w_\bullet + w_\circ)$. An analytical characterization of it is provided in Appendix B.3: for the moment, let us simply notice that the definition provided above works for the values $0 < l_\bullet, l_\circ < L$. The position $f(l_\bullet, l_\circ, w_\bullet, w_\circ) \equiv \text{MEH}(l_\bullet, l_\circ, w_\bullet, w_\circ | V, V_\bullet, V_\circ, L, W)$ induces the definition of the *enhanced bimodular surprise*

$$\mathscr{E}_{/\!/} = \sum_{l_\bullet \geq l_\bullet^*} \sum_{l_\circ \geq l_\circ^*} \sum_{w_\bullet \geq w_\bullet^*} \sum_{w_\circ \geq w_\circ^*} f(l_\bullet, l_\circ, w_\bullet, w_\circ). \tag{3.42}$$

Notice that $l_\bullet$ and $l_\circ$ are 'naturally' bounded by $V_\bullet$ and $V_\circ$: still, their sum cannot exceed L; analogously, $w_\bullet - l_\bullet$ and $w_\circ - l_\circ$ are 'naturally' bounded by $W - L$: still, their sum itself cannot exceed $W - L$.

As for its binomial counterpart, the expression of the MEH can be rearranged in a term-by-term fashion, in such a way that the module-specific binomial coefficients can be grouped together. Upon doing so, it becomes clearer that the MEH counts the number of ways $w_\bullet - l_\bullet$ links can be placed on top of the $l_\bullet$ binary links characterizing the connectance of the $\bullet$ module, times the number of ways $w_\circ - l_\circ$ links can be placed on top of the $l_\circ$ binary links characterizing the connectance of the $\circ$ module, times the number of ways the remaining $W - (w_\bullet + w_\circ) - (L - (l_\bullet + l_\circ))$ links can be placed on top of the $L - (l_\bullet + l_\circ)$ binary links characterizing the connectance of the third module.

Asymptotic results

The enhanced bimodular surprise admits an asymptotic expression as well, i.e.

$$\mathscr{E}_{/\!/} \simeq \sum_{l_\bullet \geq l_\bullet^*} \sum_{l_\circ \geq l_\circ^*} \sum_{w_\bullet \geq w_\bullet^*} \sum_{w_\circ \geq w_\circ^*} F(l_\bullet, l_\circ, w_\bullet, w_\circ) \left[\frac{\mathrm{BF}(V, L, W, p, r)}{\prod_{i=\bullet,\circ,\top} \mathrm{BF}(V_i, l_i, w_i, p_i, r_i)} \right] \tag{3.43}$$

that can be recovered by Stirling-approximating the binomial coefficients entering into the definition of $\mathscr{E}_{/\!/}$. While the expression 'BF' denotes a Bose-Fermi probability mass function [58], the parameters appearing in eq. (3.43) read $p = \frac{L}{V}$, $p_i = \frac{l_i}{V_i}$ and $r = \frac{W-L}{W-1}$, $r_i = \frac{w_i - l_i}{w_i - 1}$; as for $\mathscr{E}$, the maximum-of-the-likelihood principle determines (approximates) the first (second) class of parameters.

The numerical coefficient multiplying the whole expression is defined as

$$F(l_\bullet, l_\circ, w_\bullet, w_\circ) = \frac{1}{(2\pi)^2} \sqrt{\frac{\sigma^2 \mu}{\prod_{i=\bullet,\circ,\top} \sigma_i^2 \mu_i}} \tag{3.44}$$

where $\sigma^2 = Vp(1-p)$, $\sigma_i^2 = V_i p_i (1-p_i)$, $\mu \simeq Lr$ and $\mu_i \simeq l_i r_i$. As observed for the other functionals, the asymptotic expression of the enhanced bimodular suprise compares the description of a networked configuration provided by the ERGM, and encoded into the expression $\mathrm{BF}(V, L, W, p, r) = p^L (1-p)^{V-L} \cdot r^{W-L} (1-r)^L$, with the description of the same configuration provided by its block-wise counterpart, i.e. the ESBM (now, defined by three - instead of two - different blocks), 'represented' by the denominator of the expression defined in eq. (3.43), i.e. $\prod_{i=\bullet,\circ,\top} \mathrm{BF}(V_i, l_i, w_i, p_i, r_i)$.

Table 7 gathers all the variants of the surprise-based formalism, illustrating both the full and the asymptotic expression for each of them. To sum up, detecting a weighted mesoscale structure implies considering the *negative* version of the probability mass function working in the corresponding binary case (e.g. moving from the hypergeometric to the negative hypergeometric one); detecting a 'bimodular' structure, instead, implies considering the *multinomial* version of the probability mass function working in the corresponding binary case (e.g. moving from the binomial hypergeometric to the multinomial one).

Full expression	Community detection	Bimodular structures detection
Binary case	$f = \frac{\prod_{i=\bullet,\circ}\binom{V_i}{l_i}}{\binom{V}{L}}$	$f = \frac{\prod_{i=\bullet,\circ,\top}\binom{V_i}{l_i}}{\binom{V}{L}}$
Weighted case	$f = \frac{\prod_{i=\bullet,\circ}\binom{V_i+w_i-1}{w_i}}{\binom{V+W-1}{W}} = \frac{\prod_{i=\bullet,\circ}\left(\!\binom{V_i}{w_i}\!\right)}{\left(\!\binom{V}{W}\!\right)}$	$f = \frac{\prod_{i=\bullet,\circ,\top}\binom{V_i+w_i-1}{w_i}}{\binom{V+W-1}{W}} = \frac{\prod_{i=\bullet,\circ,\top}\left(\!\binom{V_i}{w_i}\!\right)}{\left(\!\binom{V}{W}\!\right)}$
Enhanced case	$f = \frac{\prod_{i=\bullet,\circ}\binom{V_i}{l_i}\binom{w_i-1}{l_i-1}}{\binom{V}{L}\binom{W-1}{W-L}}$	$f = \frac{\prod_{i=\bullet,\circ,\top}\binom{V_i}{l_i}\binom{w_i-1}{l_i-1}}{\binom{V}{L}\binom{W-1}{W-L}}$

Asymptotic expression	Community detection	Bimodular structures detection
Binary case	$f \propto \frac{\text{Ber}(V,L,p)}{\prod_{i=\bullet,\circ}\text{Ber}(V_i,l_i,p_i)}$	$f \propto \frac{\text{Ber}(V,L,p)}{\prod_{i=\bullet,\circ,\top}\text{Ber}(V_i,l_i,p_i)}$
Weighted case	$f \propto \frac{\text{Geo}(V,W,q)}{\prod_{i=\bullet,\circ}\text{Geo}(V_i,w_i,q_i)}$	$f \propto \frac{\text{Geo}(V,W,q)}{\prod_{i=\bullet,\circ,\top}\text{Geo}(V_i,w_i,q_i)}$
Enhanced case	$f \propto \frac{\text{BF}(V,L,W,p,r)}{\prod_{i=\bullet,\circ}\text{BF}(V_i,l_i,w_i,p_i,r_i)}$	$f \propto \frac{\text{BF}(V,L,W,p,r)}{\prod_{i=\bullet,\circ,\top}\text{BF}(V_i,l_i,w_i,p_i,r_i)}$

Table 7: Table illustrating all the generalizations of the surprise-based formalism proposed in the present thesis and showing both the full and the asymptotic expression of each probability mass function - with the only exception of the numerical coefficient characterizing each asymptotic expressions. To sum up, detecting a weighted mesoscale structure implies considering the *negative* version of the probability mass function working in the corresponding binary case (e.g. moving from the hypergeometric to the negative hypergeometric one); detecting a 'bimodular' structure, instead, implies considering the *multinomial* version of the probability mass function working in the corresponding binary case (e.g. moving from the binomial hypergeometric to the multinomial one). Upon employing the multiset notation, a nice formal symmetry can be recovered between the purely binary and the purely weighted cases.

3.6 Results

The previous sections have been devoted to the description of the surprise-based formalism for detecting a number of mesoscale structures; let us, now, test it on a bunch of synthetic and real-world configurations.

3.6.1 A consistency check: is surprise a proper p-value?

Let us start by checking the consistency of our surprise-based formalism. Since we have rephrased the problem of detecting any mesoscale structure into an exact significance test, the limitations of our formalism are the same ones affecting the tests of the kind. More precisely, any significance test is characterized by a parameter known as *type I error rate* and usually denoted with α: it quantifies the percentage of times the considered test provides a *false positive*. In other words, any statistical test is known to 'fail'; in our case, this amounts at recognizing that a significant mesoscale structure can be detected, *in case there is none*, a percentage α of the times: this number can be kept 'small' by adjusting the threshold of the p-value of the corresponding test - e.g. by deeming its response as significant in case it is less than 0.05.

Remarkably, the aforementioned behavior can be explicitly tested for each of the cases considered above. Whenever exercises like these are carried out, it is of utmost importance to be as clear as possible about the null hypothesis tested: here, we aim at testing whether *a given partition* is significant or not when the null hypothesis is true. For the sake of illustration, let us consider the problem of detecting communities on binary networks: as we learnt from the asymptotic expression of $\mathscr{S}$, it compares the description of a network provided by the RGM with that of the same network provided by the SBM, thus suggesting the RGM as the model playing the role of H_0. Hence, let us generate many networks from the RGM and, for each of them, let us impose a partition - the same for each sampled configuration, over which our SBM is tuned; finally, let us calculate $\mathscr{S}$ on each of them.

The results of such an experiment are shown in fig. 5: over 10.000 networks sampled from the RGM, whose only parameter has been set to

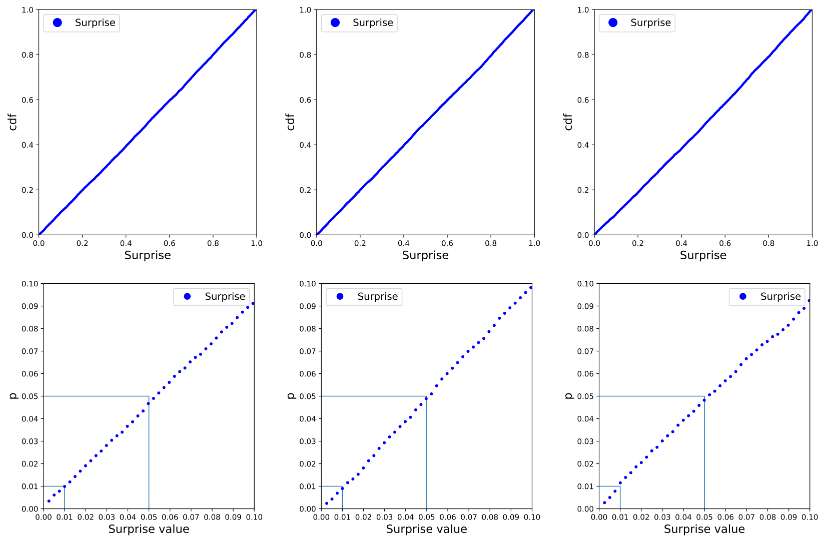


Figure 5: Consistency check of our surprise-based formalism in the simplest case of community detection on binary networks. Any exact statistical test is characterized by a parameter, usually denoted with α and known as the type I error rate, that quantifies the percentage of times the test provides a false positive: it can be kept 'low' by adjusting the threshold of the p-value of the corresponding test, e.g. by deeming its response as significant in case it is less than 0.05. Over 10.000 networks sampled by the RGM with $p = 0.2$, a given planted partition - from left to right: two, three and four clusters with different dimensions - is indeed recovered as significant 5% of the times. Notably, since the three upper panels show the CDFs of the empirical values of $\mathscr{S}$, they also provide an information about the distribution of the latter, which is uniform over the unit interval.

$p = 0.2$, the (given) planted partition - i.e. two, three and four clusters with different dimensions - is, indeed, recovered as significant 5% of the times; notice that such a result holds true irrespectively of the details of the partition imposed on the sampled configurations.

We have also repeated such an experiment for the problem of detecting communities on weighted networks: as fig. 6 shows, the same results are recovered, i.e. over 10.000 networks sampled from the WRGM, whose only parameter has been set to $q = 0.2$, a (given) planted partition

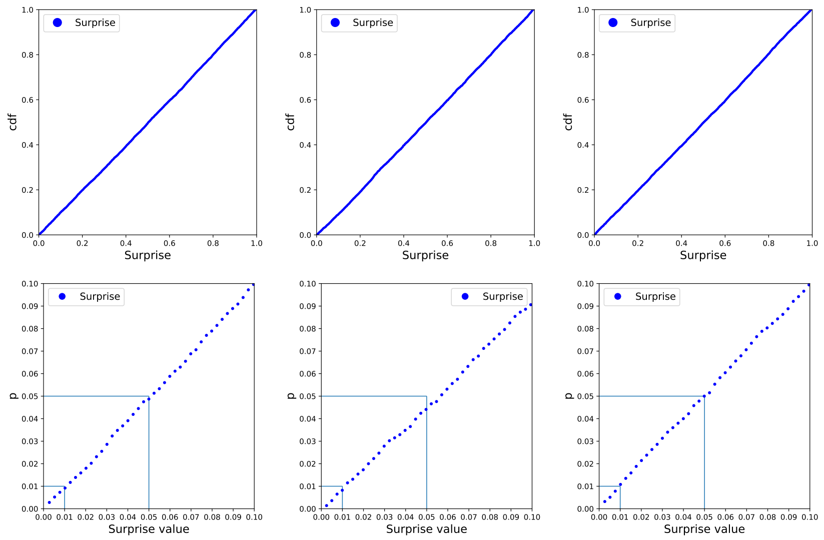


Figure 6: Consistency check of our surprise-based formalism in the case of community detection on weighted networks. As in the binary case, the percentage of type I error rate can be kept 'low' by adjusting the threshold of the p-value of the corresponding test, e.g. by deeming its response as significant in case it is less than 0.05. Over 10.000 networks sampled by the WRGM with $q = 0.2$, a given planted partition - from left to right: two, three and four clusters with different dimensions - is indeed recovered as significant 5% of the times. Notably, since the three upper panels show the CDFs of the empirical values of $\mathscr{W}$, they also provide an information about the distribution of the latter, which is uniform over the unit interval.

of two, three and four clusters with different dimensions is recognized as significant 5% of the times; again, such a result holds true irrespectively of the details of the partition imposed on our sampled configurations.

Our experiments tell us something deeper about the behavior of $\mathscr{S}$ and $\mathscr{W}$: since each upper panel in figs. 6 and 7 shows the cumulative distribution function (CDF) of the empirical values of $\mathscr{S}$ and $\mathscr{W}$, we learn that the latter ones are distributed *uniformly* over the unit interval, i.e. $\mathscr{S} \sim \mathrm{U}[0,1]$ and $\mathscr{W} \sim \mathrm{U}[0,1]$ - an evidence further confirming that surprise indeed behaves like the p-value of an exact significance test.

Checking the behavior of the multivariate versions of our hypergeometric distributions is, instead, much more difficult, the reason lying in the evidence that the null hypothesis is, now, a multivariate one and few results about the behavior of multivariate p-values are known. Still, we can say something about the behavior of the *marginal p-values*; for the sake of illustration, let us consider the ones induced by $\mathscr{S}_{/\!/}$ and $\mathscr{W}_{/\!/}$, i.e.

$$\mathscr{S}_{/\!/}^{(\bullet)} = \sum_{l_\bullet \geq l_\bullet^*} \sum_{l_\circ} f(l_\bullet, l_\circ) = \sum_{l_\bullet \geq l_\bullet^*} f(l_\bullet), \tag{3.45}$$

$$\mathscr{S}_{/\!/}^{(\circ)} = \sum_{l_\circ \geq l_\circ^*} \sum_{l_\bullet} f(l_\bullet, l_\circ) = \sum_{l_\circ \geq l_\circ^*} f(l_\circ) \tag{3.46}$$

and

$$\mathscr{W}_{/\!/}^{(\bullet)} = \sum_{w_\bullet \geq w_\bullet^*} \sum_{w_\circ} f(w_\bullet, w_\circ) = \sum_{w_\bullet \geq w_\bullet^*} f(w_\bullet), \tag{3.47}$$

$$\mathscr{W}_{/\!/}^{(\circ)} = \sum_{w_\circ \geq w_\circ^*} \sum_{w_\bullet} f(w_\bullet, w_\circ) = \sum_{w_\circ \geq w_\circ^*} f(w_\circ) \tag{3.48}$$

respectively. As evident from the formulas above, marginal distributions of multivariate, hypergeometric distributions are hypergeometric themselves: hence, the behavior of our marginal p-values is expected to match the one observed in the univariate cases, leading to recover $\mathscr{S}_{/\!/}^{(\bullet)} \sim \mathrm{U}[0,1]$, $\mathscr{S}_{/\!/}^{(\circ)} \sim \mathrm{U}[0,1]$, $\mathscr{W}_{/\!/}^{(\bullet)} \sim \mathrm{U}[0,1]$ and $\mathscr{W}_{/\!/}^{(\circ)} \sim \mathrm{U}[0,1]$. Analogously for the enhanced surprise.

Our findings can be also described from a different perspective, i.e. by answering the question *what is the expected value of the surprise on a network sampled from the RGM ensemble?* To answer this question let us focus on the simplest case of detecting communities on binary networks and consider that, irrespectively from the partition that is planted on the sampled configuration, the expected number of links will be $\langle L \rangle = pV$ while the expected number of 'internal' links will be $\langle l_\bullet^* \rangle = pV_\bullet$, where p is the parameter defining the RGM; hence, the surprise becomes

$$\mathscr{S} = \sum_{l_\bullet \geq \langle l_\bullet^* \rangle} \frac{\binom{V_\bullet}{l_\bullet}\binom{V-V_\bullet}{\langle L \rangle - l_\bullet}}{\binom{V}{\langle L \rangle}} = \sum_{l_\bullet \geq pV_\bullet} \frac{\binom{V_\bullet}{l_\bullet}\binom{V-V_\bullet}{pV - l_\bullet}}{\binom{V}{pV}} \tag{3.49}$$

(to be fully consistent we should have rounded both the value of $\langle L \rangle$ and that of $\langle l_\bullet^* \rangle$ to the nearest integer but our conclusions are still valid). In order to evaluate the expression above, let us consider that the expected value of the hypergeometric distribution defining the surprise reads

$$\langle l_\bullet \rangle = \langle L \rangle \frac{V_\bullet}{V} = pV_\bullet \tag{3.50}$$

the validity of the first passage resting upon the evidence that the network is generated via the RGM. Hence, the surprise becomes a sum over the values $l_\bullet \geq \langle l_\bullet \rangle$, i.e. those that are more extreme than the average. Since the hypergeometric distribution is peaked around its average, the result above suggests that a sum over values that are larger than the average encodes half of the probability mass, i.e. $\simeq \frac{1}{2}$: as a consequence, the expected value of surprise, on a network generated via the RGM, amounts at $\simeq \frac{1}{2}$, irrespectively from the partition that is planted on the sampled configuration - its fluctuations being compatible with those of a uniform distribution whose support coincides with the unit interval.

3.6.2 Comparing surprise with the binary modularity

For the sake of comparison, let us consider the modularity function for the problem of community detection on binary, undirected networks. Since the asymptotic binary suprise compares the description of a network configuration provided by the RGM with the description of the same configuration provided by the SBM, hereby we will consider the definition of modularity whose benchmark model is provided by the RGM. Formally, this is achieved by substituting $p_{ij} = \frac{L}{V} = \frac{2L}{N(N-1)}$, $\forall\, i < j$ in eq. (3.1):

$$Q = \frac{1}{2L}\sum_{i\neq j} a_{ij}\delta_{c_i,c_j} - \frac{1}{2L}\sum_{i\neq j} p\delta_{c_i,c_j} \tag{3.51}$$

$$= \frac{1}{2L}\sum_{c}[2l_c^* - pN_c(N_c-1)] \tag{3.52}$$

$$= \frac{1}{2L}\sum_{c}\left[2l_c^* - 2L\frac{N_c(N_c-1)}{N(N-1)}\right] \tag{3.53}$$

$$= \left(\frac{l_\bullet^*}{L} - \frac{V_\bullet}{V}\right) \tag{3.54}$$

where N_c and l_c^* indicate the size of community c and the number of links *within* it, respectively - naturally, $l_\bullet^* = \sum_c l_c^*$ and $V_\bullet = \sum_c N_c(N_c - 1)$. This calculation clarifies that a positive value of Q implies that the 'internal' probability of connection $p_\bullet$ is larger than the one 'predicted' by the RGM, i.e.

$$Q \geq 0 \Longrightarrow p_\bullet = \frac{l_\bullet^*}{V_\bullet} \geq \frac{L}{V} = p \tag{3.55}$$

while a null modularity value implies that $p_\bullet$ matches the only parameter defining the RGM. The result above can be also restated as follows: a null modularity value points out that the mesoscale structure of the configuration at hand *has not been generated* by a SBM (or, equivalently, *does not need* a SBM to be explained). Notice, however, that the modularity provides no indication about the statistical significance of the recovered partition.

3.6.3 Comparing surprise with the weighted modularity

A similar conclusion can be reached upon considering the modularity function for community detection on weighted, undirected networks, defined as

$$Q = \frac{1}{2W}\sum_{i\neq j}(w_{ij} - \langle w_{ij}\rangle)\delta_{c_i,c_j} \tag{3.56}$$

where we have employed the Weighted Random Graph Model (WRGM) as a benchmark, according to which $\langle w_{ij}\rangle_{\text{WRGM}} = \frac{W}{V} = \frac{2W}{N(N-1)}$, $\forall\, i < j$ [105]. From the definition of weighted modularity provided above, it follows that

$$Q = \frac{1}{2W}\sum_{i\neq j} w_{ij}\delta_{c_i,c_j} - \frac{1}{2W}\sum_{i\neq j}\langle w_{ij}\rangle_{\text{WRGM}}\delta_{c_i,c_j} \tag{3.57}$$

$$= \frac{1}{2W}\sum_{c}[2w_c^* - \langle w_{ij}\rangle_{\text{WRGM}} N_c(N_c-1)] \tag{3.58}$$

$$= \frac{1}{2W}\sum_{c}\left[2w_c^* - 2W\frac{N_c(N_c-1)}{N(N-1)}\right] \tag{3.59}$$

$$= \left(\frac{w_\bullet^*}{W} - \frac{V_\bullet}{V}\right) \tag{3.60}$$

where N_c and w_c^* indicate the number of nodes of community c and the weight of links *within* community c, respectively - naturally, $w_\bullet^* = \sum_c w_c^*$. As in the binary case, the calculation above clarifies that a positive value of Q implies that the expected weight of any 'internal' link is larger than the one predicted by the WRGM, i.e.

$$Q \geq 0 \Longrightarrow \langle w_{ij}\rangle_{\text{WSBM}} = \frac{w_\bullet^*}{V_\bullet} \geq \frac{W}{V} = \langle w_{ij}\rangle_{\text{WRGM}}; \tag{3.61}$$

a null modularity value, on the other hand, implies that $\langle w_{ij}\rangle_{\text{WSBM}}$ coincides with the expectation coming from the WRGM. Analogously to the binary case, the result above can be also restated as follows: a null modularity value points out that the mesoscale structure of the configuration at hand *has not been generated* by a WSBM (or, equivalently, *does not need* a WSBM to be explained). Again, however, the modularity provides no indication about the statistical significance of the recovered partition.

3.6.4 Comparing mesoscale structures detection methods

The previous subsections have been devoted to check the consistency of our surprise-based formalism and reveal the 'statistical flaws' of other approaches. Let us now consider two popular algorithms for mesoscale structures detection, i.e. modularity maximization (now, Q has been considered in its 'full' definition, i.e. $\langle a_{ij}\rangle = p_{ij} = \frac{k_i k_j}{2L}$, $\forall\, i < j$ for binary, undirected configurations) and Infomap [36], and carry out more systematical comparisons between the former ones and the surprise. Upon doing so, we are able to compare one algorithm per class, i.e. modularity for the first class, surprise for the second class and Infomap for the third class.

Evaluating the quality of a given partition

To this aim, we have focused on different kinds of benchmarks, i.e. classes of synthetic networks with well-defined planted partitions, the aim being that of inspecting the goodness of a given algorithm in recovering the imposed partition. As an indicator of the goodness of the partition retrieved by each algorithm, we have followed [89] and employed three different indices.

The first one is the *normalized mutual information* (NMI), defined as

$$\overline{I}(X,Y) = \frac{2I(X,Y)}{H(X)+H(Y)} \tag{3.62}$$

where partitions X and Y are compared, $H(X) = -\sum_x f_x \ln f_x$ and $f_x = \frac{n_x}{n}$ is the fraction of nodes assigned to the cluster labeled with x; analogously, for the partition Y. The term $I(X,Y) = \sum_x \sum_y f_{xy} \ln\left(\frac{f_{xy}}{f_x f_y}\right)$ is the 'proper' *mutual information* and $f_{xy} = \frac{n_{xy}}{n}$ is the fraction of nodes assigned to cluster x in partition X *and* to cluster y in partition Y. Naturally, $\overline{I}(X,Y)$ equals 1 if the partitions are identical and 0 if the partitions are independent.

The second index we have considered is the *adjusted Rand index* (ARI), defined as

$$\text{ARI} = \frac{TP+TN-\langle TP+TN\rangle}{TP+FP+TN+FN-\langle TP+TN\rangle} \tag{3.63}$$

and representing a sort of accuracy[5] 'corrected' by a term that quantifies

[5]The number of true positives (TP) is the number of pairs of nodes being in the same community both in the considered and in the reference partition; the number of false pos-

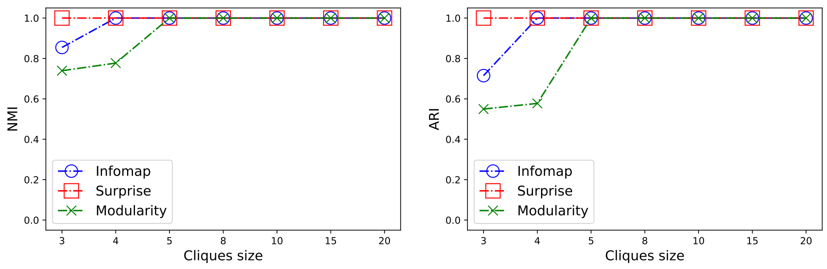


Figure 7: Performance of modularity, surprise and Infomap in correctly identifying the partitions induced by seven, different, 'homogeneous' rings of cliques. For each configuration, twenty cliques have been considered; the size of the latter ones is left to vary as follows: K_3, K_4, K_5, K_8, K_{10}, K_{15}, K_{20}. While surprise always recovers the planted partition, modularity misses the partitions with K_3 and K_4 and Infomap misses the partition with K_3 - a result that may be a consequence of the resolution limit which is known to affect the last two algorithms.

the agreement between the reference partition and a random partition - the term 'random' referring to the Permutation Model; equivalently, the closer the ARI to 0, the more 'random' the provided partition.

The third index we have considered is the *adjusted Wallace index* (AWI), defined as

$$\text{AWI} = \frac{TP - \langle TP \rangle}{TP + FP - \langle TP \rangle} \tag{3.64}$$

and representing a sort of 'corrected' positive predicted value. Again, the closer the AWI to 0, the more 'random' the provided partition.

Testing algorithms on synthetic, modular benchmarks

First, let us inspect the performance of modularity, surprise and Infomap to detect cliques arranged in a ring. Specifically, we have considered 7,

itives (FP) is the number of pairs of nodes being in the same community in the considered partition but in different communities in the reference partition; the number of true negatives (TN) is the number of pairs of nodes being in the same community neither in the considered nor in the reference partition; the number of false negatives (FN) is the number of pairs of nodes being in the same community in the reference partition but not in the considered partition (see [89] for more details).

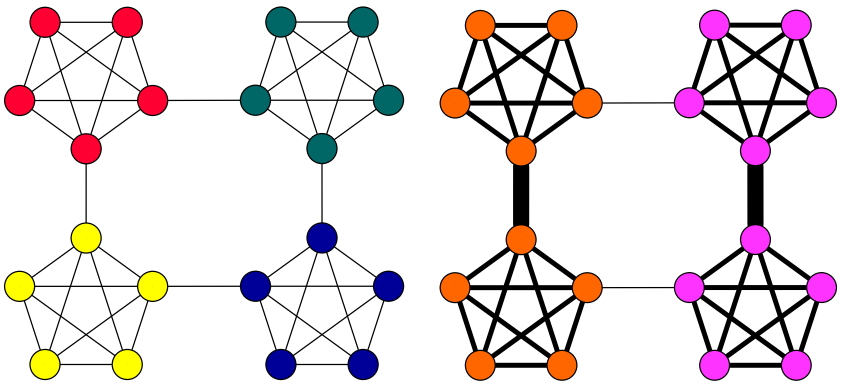


Figure 8: Comparison between the 'ring of binary cliques' and the 'ring of weighted cliques' cases. The result according to which surprise minimization is able to discriminate the cliques linked in a ring-like fashion changes once the weights come into play: in fact, as the weight of the links connecting any two cliques is risen, the algorithm reveals as 'communities' pairs of tightly-connected cliques.

different ring-like configurations, each one linking 20 binary cliques (i.e. K_3, K_4, K_5, K_8, K_{10}, K_{15}, K_{20}). As fig. 7 reveals, surprise always recovers the planted partition; on the other hand, modularity maximization leads to miss the partitions with K_3 and K_4 and Infomap misses the partition with K_3, a result that may be a consequence of the resolution limit, affecting both the aforementioned algorithms.

Let us now ask us if the presence of weights affects the detection of mesoscale structures. Generally speaking, the answer is yes, as the comparison between the 'ring of binary cliques' and the 'ring of weighted cliques' cases, depicted in fig. 8, shows. In particular, the result according to which surprise minimization is able to discriminate the interlinked cliques changes once the weight of the links connecting any two cliques is risen: in fact, this leads the algorithm to reveal as 'communities' two tightly-connected pairs of cliques, now. We explicitly notice that the results shown in fig. 8 also depend on the relative magnitude of the weights of the intra-cliques and of the inter-cliques links: however, as long as the inter-cliques weight is up to two orders of magnitude larger than the intra-cliques one, it holds true.

In order to expand the set of comparisons, we have focused on two different kinds of well-established benchmarks, i.e. the Lancichinetti-Fortunato-Radicchi (LFR) [106] one and the Aldecoa's 'relaxed-caveman' (RC) one [32]. Figure 9 shows two examples of the benchmarks used in the present paper to compare modularity, surprise and Infomap: LFR, on the left, and RC, on the right.

The LFR benchmark is a special case of the planted-partition model, in which groups have different sizes and nodes have different degrees - hence constituting a refinement of the GN benchmark, where groups have equal size and nodes have the same expected degree [106]. When binary, undirected configurations are considered, the degrees of nodes are distributed according to a power-law with exponent τ_1 while the sizes of communities are distributed according to a power-law with exponent τ_2. Once the sizes of communities have been drawn each node 'receives' its own degree, say k_i: the percentage of these links connecting node i with other internal nodes is, then, chosen to be $(1-\mu_t)k_i$, with μ_t playing the role of mixing parameter that controls for the sharpness of the planted partition. For binary, directed configurations, μ_t refers to in-degrees, which are distributed according to a power-law while the out-degrees are kept constant for all nodes; the other input parameters, instead, are the same used for undirected configurations.

Results on specific implementations of the LFR benchmark are shown in fig. 10. Four curves are shown for both binary and weighted networks: in the binary case, we consider networks of different size (blue and red: 1.000 nodes; green and black: 5.000 nodes) and with different planted communities ('S' stands for 'small': communities have between 10 and 50 nodes; 'B' stands for 'big': communities have between 20 and 100 nodes); in the weighted case, instead, all networks have the same size (5.000 nodes) but differ for the value of the binary mixing parameter (blue and red: $u_t = 0.5$; green and black: $u_t = 0.8$) and for the kind of planted communities ('S' stands for 'small': communities have between 10 and 50 nodes; 'B' stands for 'big': communities have between 20 and

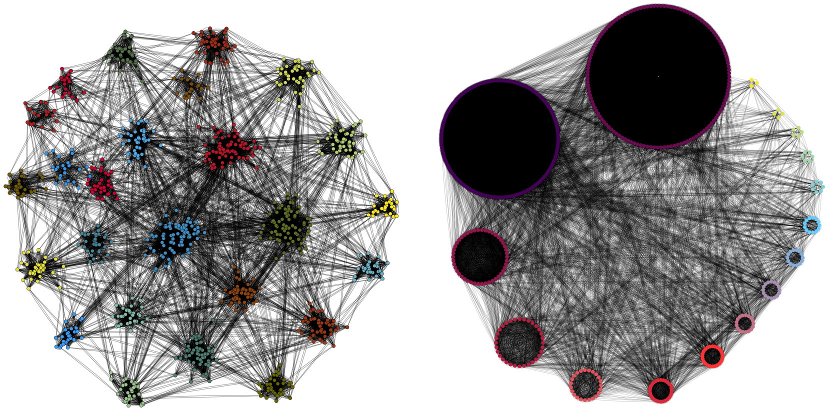


Figure 9: Examples of the benchmarks used in the present paper to compare modularity, surprise and Infomap: Lancichinetti-Fortunato-Radicchi benchmark (LFR, left panel), constituted by 1.000 nodes arranged in 'big' communities and with mixing coefficient $\mu_t = 0.1$; Aldecoa's 'relaxed-caveman' benchmark (RC, right panel), constituted by 16 communities whose size is distributed according to a power-law and with 'degradation' coefficient $p = 0.1$.

100 nodes).

When focusing on binary, undirected networks, we have considered $\tau_1 = -2$ and $\tau_2 = -1$ (the average degree is 20 and the maximum degree is 50). When binary, directed configurations are considered, μ_t refers to in-degrees, which are distributed according to a power-law while the out-degrees are kept constant for all nodes; the other input parameters, instead, are the same used for undirected configurations.

When weighted, undirected configurations are considered, an additional mixing parameters is needed, i.e. μ_w, accounting for the percentage of a node strength to be distributed on the links that connect it to the nodes outside its own community; the exponent of the strength distribution has been set to 1.5 for all realizations considered here. When weighted, directed configurations are considered, μ_w refers to in-strengths.

Infomap is, generally speaking, a strong performer; as evident upon looking at the first row, however, its performance decreases abruptly as the mixing parameter exceeds a threshold value that depends on the particular setting of the LFR benchmark. Modularity, instead, seems to be more robust (i.e. its performance 'degrades' less rapidly as μ_t increases) although the resolution limit manifests itself when configurations with small communities are considered. Overall, the performance of surprise seems to constitute a good compromise between the robustness of modularity and the steadily high accuracy of Infomap. We would also like to stress that surprise competes with modularity although it employes much less information than the latter: in fact, while the benchmark employed by modularity coincides with the (sparse version of the) Configuration Model - hence, encodes the information on the entire degree sequence - surprise compares the RGM with the SBM, hence employing the information on the link density, both in a global and in a block-wise fashion.

Surprise becomes the best performer when binary, directed configurations are considered - see the second row of fig. 10: while the performance of modularity starts decreasing as soon as the value of μ_t is risen and $\text{NMI}_{Infomap} \simeq 0$ when μ_t crosses the value of 0.6, the performance of surprise 'degrades' much more slowly - in fact, for some instances of the LFR benchmark, it achieves a large value of NMI even for values $\mu_t \geq 0.8$.

Let us now comment on the performance of our algorithms when weighted configurations are considered (to be noticed that we have kept one of the two parameters fixed and studied the dependence of NMI and ARI on the other: specifically, we have frozen the topological mixing parameter and studied the dependence of the results on μ_w, thus inspecting the performance of our algorithms as the weights are redistributed on a fixed topology): the results are, again, shown in fig. 10. Infomap is, again, a strong performer although its performance keeps decreasing abruptly as μ_w exceeds a threshold value depending on the particular setting of the LFR benchmark; modularity, instead, performs worse than in the binary case although it is still more robust than Infomap. Although 'degrading' less sharply than Infomap, the performance of the purely weighted surprise seems to be the worst, here; on the other hand,

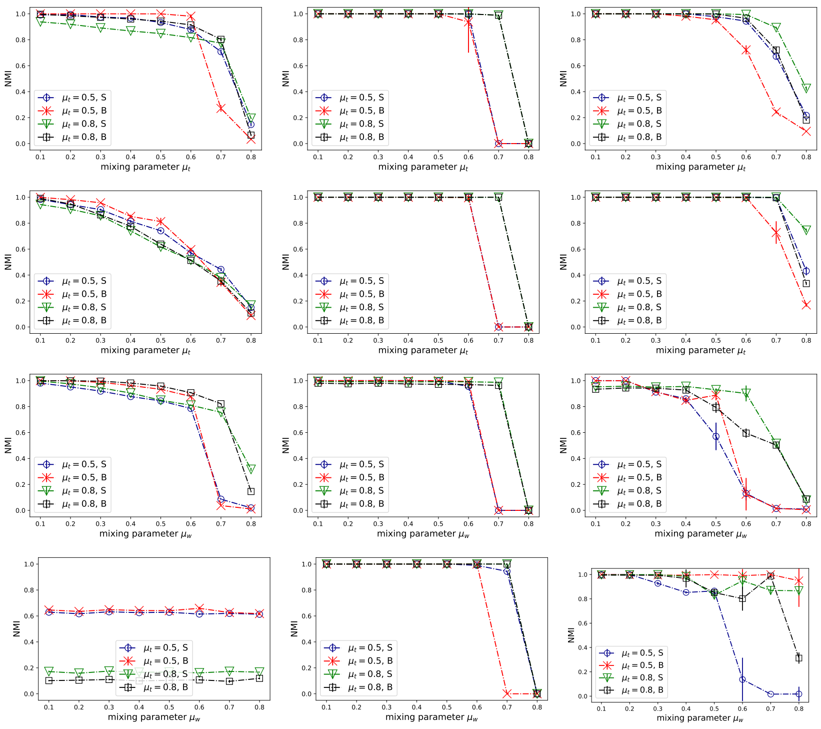


Figure 10: Comparison of three different algorithms for community detection on the LFR benchmark, i.e. modularity maximization (left column), Infomap (central column) and surprise minimization (right column): the chosen configurations are binary undirected networks (first row), binary directed networks (second row), weighted undirected networks (third row) and weighted directed networks (fourth row). The trend of the NMI, plotted as a function of the mixing parameters, reveals Infomap to be a strong performer even if its performance decreases abruptly as the mixing parameters exceed a threshold value that depends on the particular setting of the benchmark; the performance of modularity, instead, 'degrades' less rapidly. The performance of surprise seems to constitute a good compromise between the robustness of modularity and the steadily high accuracy of Infomap.

the enhanced surprise outperforms the competing algorithms for intermediate values of the topological mixing parameter, irrespectively from the size of the communities: in fact, $\text{NMI}_{surprise} = 1$ even for $\mu_w = 0.8$.

Similar considerations hold true when weighted, directed configurations are considered (with the only difference that, now, modularity steadily performs worse than the other algorithms, except for the largest values of the mixing parameter). As for the binary cases, surprise competes with modularity although it employes much less information than the latter: in fact, while the benchmark employed by modularity now coincides with the (sparse version of the) Weighted Configuration Model - hence, encodes the information on the entire strength sequence - surprise compares the WRGM with the WSBM, hence employing the information on the magnitude of the total weight, both in a global and in a block-wise fashion.

Let us now consider the RC benchmark. It consists of 512 nodes, grouped in 16 communities arranged in a ring-like fashion and whose sizes obey a power-law whose exponent has been set to 1.8; the smallest community is composed by 3 nodes. Such a configuration is progressively 'degraded' according to the following mechanism: first, a percentage p of links is randomly selected and removed; afterwards, a percentage p of links is randomly selected and rewired. In other words, a single 'degradation' parameter p drives the evolution of the initial ring-of-cliques towards a progressively less-defined, clustered configuration.

Results on specific implementations of the RC benchmark are shown in fig. 11: surprise outperforms both competing algorithms across the entire domain of the 'degradation' parameter p. More specifically, while modularity 'degrades' slowly as the value of p is risen, Infomap 'degrades' abruptly as $p \geq 0.4$. Hence, for small values of such a parameter, Infomap outperforms modularity; on the other hand, for large values of p, modularity outperforms Infomap (although both $\text{NMI}_{modularity}$ and $\text{ARI}_{modularity}$ achieve a value which is around 0.6, i.e. already far from the maximum). Interestingly, for small values of p, the performance of Infomap and that of surprise overlap, both achieving NMI and ARI values which are very close to 1: as p crosses the value of 0.4, however, the two trends become increasingly different with Infomap being outperformed by modularity which is, in turn, outperformed by surprise.

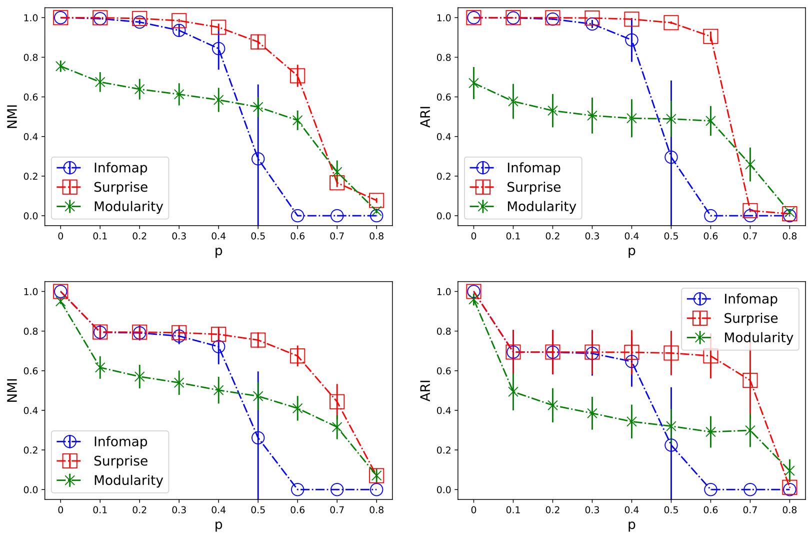


Figure 11: Comparison of three different algorithms for community detection on the RC benchmark, i.e. modularity maximization, Infomap and surprise minimization. The chosen configurations are binary, undirected (first row) and directed (second row) networks, consisting of 512 nodes, grouped in 16 communities arranged in a ring-like fashion and whose sizes obey a power-law whose exponent has been set to 1.8; the smallest community is composed by 3 nodes. This initial configuration is progressively 'degraded' according to the following mechanism: first, a percentage p of links is randomly selected and removed; afterwards, a percentage p of links is randomly selected and rewired. The trend of the NMI, plotted as a function of the (single) 'degradation' parameter p, driving the evolution of the initial ring-of-cliques towards a progressively less-defined, clustered configuration, reveals surprise to outperform both modularity and Infomap. From a more general perspective, these results confirm what has been already observed elsewhere, i.e. that the best-performing algorithms on the LFR benchmarks often perform poorly on the RC benchmarks and vice versa.

From a more general perspective, these results confirm what has been already observed elsewhere [41], i.e. that the best-performing algorithms on the LFR benchmarks often perform poorly on the RC benchmarks and vice versa.

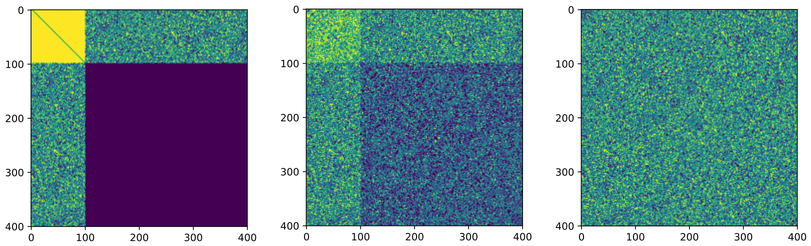


Figure 12: Benchmark for testing surprise on the recovery of core-periphery structures: we progressively 'degrade' an initial configuration, defined by 1) a completely connected core, 2) an empty periphery, 3) an intermediate part whose link density amounts at $p_{cp} = 0.5$. Such a configuration is 'degraded' by progressively filling the periphery and emptying the core. This is achieved by 1) considering all peripherical node pairs and link them with probability q; 2) considering all core node pairs and keep them linked with probability $1 - q$: varying q in the interval $[0, p_{cp}]$ allows us to span a range of configurations starting with the Borgatti-Everett one and ending with an Erdös-Rényi one.

Testing algorithms on synthetic, bimodular benchmarks

Let us now inspect the performance of surprise in recovering binary, 'bimodular' structures. To this aim, we have defined a novel benchmark (see fig. 12) mimicking the philosophy of the RC one, i.e. progressively 'degrading' an initial, well-defined configuration:

- let us consider N_c core nodes and N_p periphery nodes. The core is completely connected (i.e. the link density of the $N_c \times N_c$ block is 1) and the periphery is empty (i.e. the link density of the $N_p \times N_p$ block is 0). So far, our benchmark is reminiscent of a core-periphery structure *à la Borgatti-Everett* (left panel of fig. 12);

- let us now focus on the topology of the $N_c \times N_p$ bipartite network embodying the connections between the core and the periphery: in particular, let us consider each entry of such an adjacency matrix and pose $a_{cp} = 1$ with probability p_{cp}. Upon doing so, such a subgraph will have a link density amounting precisely at p_{cp} (central panel of fig. 12);

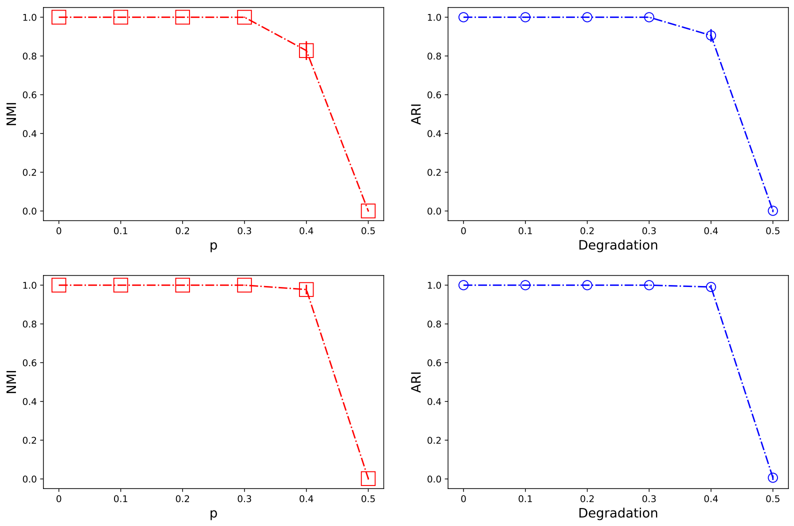


Figure 13: Performance of surprise on the recovery of core-periphery structures. The considered benchmark is mimicking the philosophy of the RC one: 'degrading' an initial, well-defined, core-periphery configuration (left column: a binary, undirected one; right column: a binary, directed one). As expected, the performance of the algorithm worsens as the 'degradation' parameter becomes closer to $p_{cp} = 0.5$; however, both the NMI and the ARI indices steadily remain very close to 1.

- let us now 'degrade' such an initial configuration, by progressively filling the periphery and emptying the core. This can be achieved by 1) considering all peripherical node pairs and link them with probability q_p; 2) considering all core node pairs and keep them linked with probability $1 - q_c$ (or, equivalently, disconnect them with probability q_c). Upon doing so, we end up with a core whose link density is precisely $1 - q_c$ and with a periphery whose link density is precisely q_p. Now, varying q_p in the interval $[0, p_{cp}]$ and q_c in the interval $[0, 1-p_{cp}]$ allows us to span a range of configurations starting with the Borgatti-Everett one and ending with an Erdös-Rényi one (right panel of fig. 12).

Specifically, here we have considered $N_c = 100$, $N_p = 300$ and $p_{cp} =$

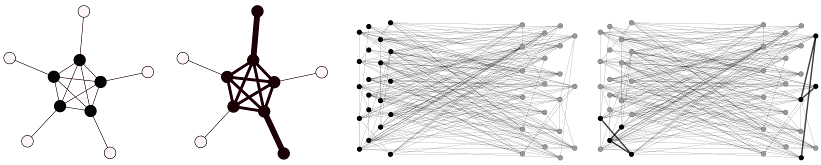


Figure 14: The presence of weights affects the detection of 'bimodular', mesoscale structures as well. In fact, rising the weight of any two links connecting the core with the periphery of a toy network allows the two nodes originally part of the periphery to be detected as belonging to the core; analogously, if a bipartite topology is modified by adding weights between some of the nodes belonging to the same layer, a core-periphery structure will be detected as significant.

0.5 - a choice allowing us to carry out the aforementioned exercise by tuning just one parameter instead of two (i.e. $q_p = q_c \equiv q$). The result of our exercise is shown in fig. 13: as expected, the performance of the surprise worsens as the 'degradation' parameter becomes closer to $p_{cp} = 0.5$; however, both the NMI and the ARI indices steadily remain very close to 1 - a result indicating that surprise optimization not only scores high under the true positives metrics - by 'keeping together' the nodes originally in the same communities - but also under the other, possible metrics.

Let us now ask us if the presence of weights affects the detection of mesoscale structures. Generally speaking, the answer is, again, yes. Let us consider a toy core-periphery network: rising the weight of any two links connecting the core with the periphery allows the two nodes originally part of the periphery to be detected as belonging to the core (see the first and the second panel of fig. 14). Analogously, if a bipartite topology is modified by adding weights between some of the nodes belonging to the same layer, $\mathscr{W}_{/\!/}$ will detect a core-periphery structure as significant, the core nodes being the ones linked by the 'heaviest' connections (see the third and the fourth panel of fig. 14).

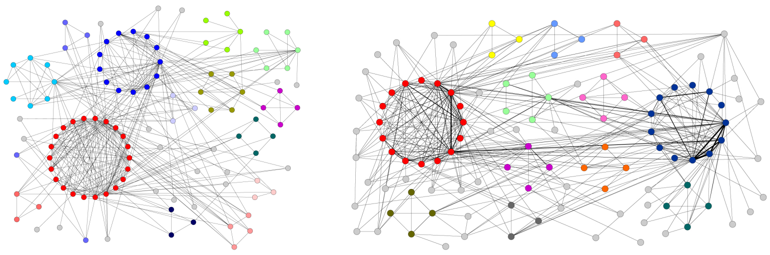


Figure 15: Application of our framework for the detection of binary and weighted communities (respectively, on the left and on the right) on the network of co-occurrences of 'Star Wars' characters [107]. Overall, link weights refine the picture provided by just considering the presence of links: in the binary case, in fact, surprise detects two major clusters, induced by the characters of Episodes I-III and by the characters of Episodes IV-IX; once weights are taken into account, these clusters merge and give origin to the cluster of heroes of Episodes IV-IX.

3.6.5 Testing surprise on real-world networks

Let us now apply our formalism to the detection of mesoscale structures in real-world networks. When coming to study such systems, particularly insightful examples are provided by social networks. To this aim, let us consider the one induced by the co-occurrences of characters within the 'Star Wars' saga (i.e. the three trilogies). As shown in fig. 15 we have both considered the binary and the weighted version of it. For what concerns the binary version of such a network, the optimization of $\mathscr{S}$ reveals the presence of two major clusters: remarkably, they are induced by the characters of Episodes I-III (e.g. Yoda, Qui-Gon, Obi-Wan, Anakin, Padme, the Emperor, Count Dooku, etc.) and by the characters of Episodes IV-IX (e.g. C-3PO, Leia, Han, Lando, Poe, Finn); a third cluster, instead, concerns the villains of Episodes VII-IX (i.e. Snoke, Kylo Ren, Phasma, Hux). Interestingly, Rey, BB-8, Maz-Kanata and other characters living on Jakku are clustered together. Quite remarkably, the interactions between the characters of Episodes IV-VI and those of Episodes VII-IX causes the former ones and the latter ones to be recovered within the same cluster. This picture is further refined once weights are taken into

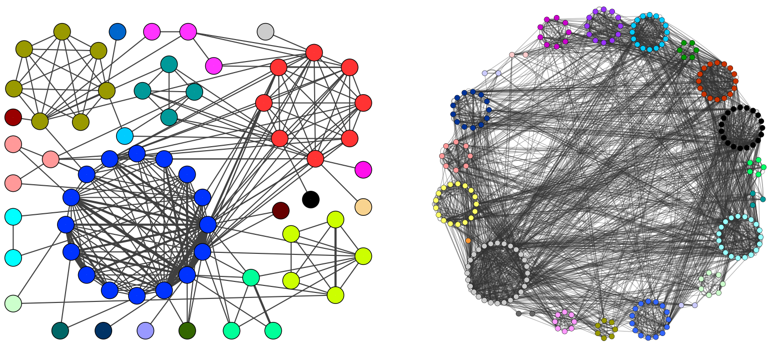


Figure 16: Application of the surprise-based framework for the detection of weighted, modular structures on two social networks: (left) weighted 'friendship' network among the terrorists involved in the train bombing of Madrid in 2004 [108], (right) weighted 'friendship' network among the residents living in an Australian University Campus [108].

account: in fact, two of the aforementioned clusters are now merged, giving origin to the cluster of heroes of Episodes IV-IX (e.g. C-3PO, Leia, Han, Lando, Poe, Finn, Rey, Maz-Kanata).

Let us now inspect the effectiveness of our framework in revealing weighted communities by considering the 'friendship' networks among the terrorists involved in the train bombing of Madrid in 2004 [108] and the one among the residents living in an Australian University Campus [108] (see fig. 16). As the optimization of $\mathscr{W}$ reveals, while fully connected subsets of nodes are considered as communities in case links have unitary weights, sparser subgraphs can be considered as communities as well whenever their inner connections are 'heavy' enough. On the other hand, the panels of figures 15 and 16 seem to confirm that one of the main limitations of surprise-like functionals is that of recovering a large number of small clusters of nodes.

Let us now compare the performance of $\mathscr{S}_{/\!/}$ and $\mathscr{W}_{/\!/}$ in order to see if, and how, the presence of weights affects the 'bimodular', mesoscale organization of networks. To this aim, let us focus on the network of co-occurrences of the characters of the novel 'Les Miserables' [108]. As shown in fig. 17, link weights indeed modify the picture provided by

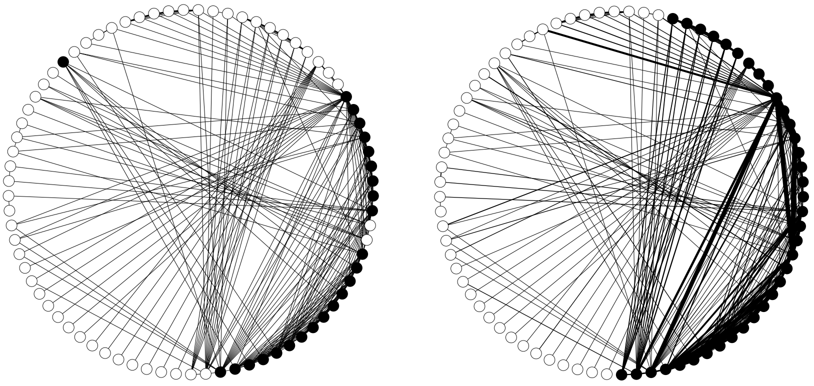


Figure 17: Application of the surprise-based framework for the detection of 'bimodular' structures on the network of co-occurrences of 'Les Miserables' characters [108] (left panel: binary version; right panel: weighted version). Link weights refine the picture provided by just considering the presence of links: the core (in black) is, in fact, constituted by the nodes connected by the 'heavier' links, irrespectively from the link density of the former one.

just considering the simple presence of links (see also [38]): the core of the weighted network is, in fact, constituted by the nodes connected by the 'heavier' links, irrespectively from the link density of the former one.

After having applied our framework to the analysis of social networks, let us move to consider financial networks. One of the most popular examples of the kind is provided by the electronic Italian Interbank Money Market (e-MID) [61], depicted in fig. 18. Notice that, for such a network, the vast majority of core links are also the 'heavier' ones, an evidence confirming a tendency that is ubiquitous in financial and economic systems, i.e. binary and weighted quantities - even at the mesoscale - are closely related.

Remarkably, the surprise-based formalism presented in this chapter can be employed in a hierarchical fashion to highlight either nested, modular or nested, 'bimodular' structures. To clarify this point, let us consider the World Trade Web (WTW) in the year 2000 as a case-study [60]. First, let us run $\mathscr{W}_{/\!/}$ to highlight the core portion of the weighted

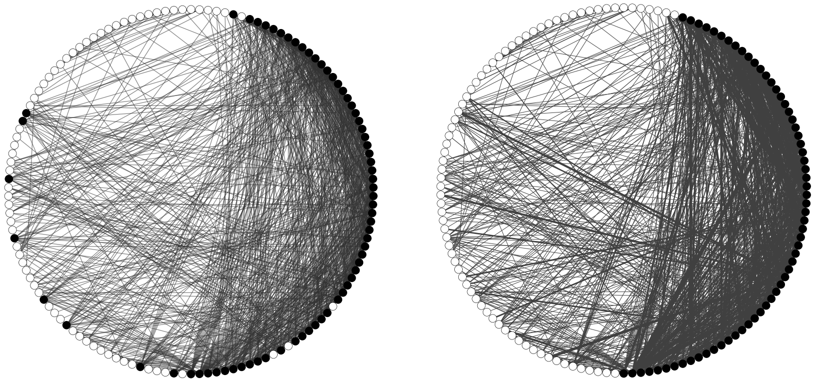


Figure 18: Application of the surprise-based framework for the detection of weighted, 'bimodular' structures on the electronic Italian Interbank Money Market (e-MID) [61], considered in the maintenance period approximately corresponding to May 2009 (left panel: binary version; right panel: weighted version). As the picture shows, the (purely binary) core links are also the 'heavier' ones - an evidence confirming an ubiquitous tendency in economic and financial systems, i.e. binary and weighted quantities are closely related.

version of such a network; as fig. 19 shows, the bipartition distinguishes countries with a large strength from those whose trade volume is low (basically, a bunch of African, Asian and South-American countries). Repeating our analysis *within* the core portion of this network allows us to discover the presence of a (statistically-significant) nested core: in fact, the second-round optimization reveals that the 'core-inside-the-core' is composed by countries such as Canada, USA, the richest European countries, Russia and China.

Let us now compare the bipartition obtained upon optimizing $\mathscr{W}_{/\!/}$ with the one obtained upon optimizing $\mathscr{E}_{/\!/}$, run in a hierarchical fashion as well. The results of our exercise are shown in fig. 19. As evident from looking at it, the enhanced surprise is more restrictive than the purely weighted one, as a consequence of constraining the degrees beside the strengths. Hence, while the first run excludes the countries with both a small degree and a small strength, the second run excludes Russia, a

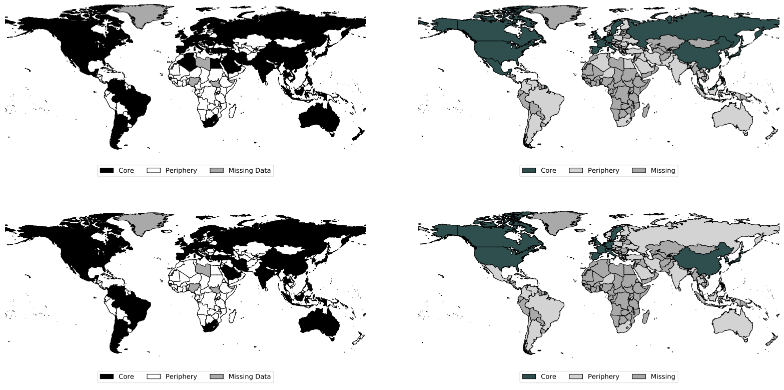


Figure 19: Application of the surprise-based framework for the detection of weighted, 'bimodular' structures on the WTW in the year 2000 [60]. Top panels: upon running $\mathscr{W}_{/\!/}$ in a hierarchical fashion, a 'core-within-the-core' is detected, revealing that the richest world countries (i.e. Canada, USA, the richest European countries, Russia and China - right panel, in dark green) constitute an even more tightly-linked cluster of nations among the ones with the largest trading activity (left panel, in black). Bottom panels: $\mathscr{E}_{/\!/}$ penalizes the countries with both a small degree and a small strength; in fact, the second run excludes Russia, a result seemingly indicating that while its strength is 'large enough' to be a member of the core, its degree is not.

result seemingly indicating that while its strength is 'large enough' to allow it to be a member of the core, its degree is not. In a sense, the optimization of $\mathscr{E}_{/\!/}$ corrects the picture provided by the optimization of $\mathscr{W}_{/\!/}$ as the core becomes less populated by 'low degree' nodes - an effect which is likely to become more evident on systems that are neither financial nor economic in nature.

Let us now run, and compare, modularity, Infomap and surprise on the bunch of real-world networks above. Table 8 sums up the results. A first observation concerns the number of detected communities: while Infomap is the algorithm producing the smallest number of clusters, surprise is the one producing the largest number of clusters - more precisely, surprise outputs *more* and *smaller* clusters than the other two methods. As a consequence, our three algorithms produce partitions with an overall small overlap, as indicated by the NMI; the ARI confirms such an observation - although indicating that the pictures provided by Infomap and surprise are (overall) more similar than those provided by modularity and surprise (and, as a consequence, by modularity and Infomap). Interestingly enough, the values of the AWI are quite large - and larger than the corresponding NMI and ARI values: since it just focuses on the percentage of true positives, a good performance under such an index indicates that the two tested algorithms 'agree' on the nodes to be clustered together (although they may not - and, in general, will not - agree on the number of communities). Hence, the discrepancy between the ARI and the AWI may be explained by the presence of statistical 'noise' (i.e. 'misclassified' pairs of nodes, although the word may not be correct as the information about the 'true' partition is not available) around the bulk of nodes to be put together.

Let us stress once more that, whenever real-world networks are considered, information about the existence of a 'true' partition is rarely available; for this reason, exercises as the one we have carried out here may be useful to gain insight on the system under study: instead of trusting just one algorithm, combining pairs of them - e.g. by considering as communities the subsets of nodes output by both - may be the right solution to overcome the limitations affecting each single method.

3.7 The 'SurpriseMeMore' package

As an additional result, we have released a comprehensive package, coded in Python, that implements all the aforementioned surprise-like variants: its name is 'SurpriseMeMore'[6] and is available on Github [109].

The development and release of an easy-to-install Python module follows the example of many of the most popular community detection algorithms for networks, [110, 111, 112]; generally speaking, in fact, the deployment of an algorithm that optimize a certain score function and

[6]The name recalls that of the package released in 2014 [33].

Binary networks	# communities								
	Modularity	Infomap	Surprise	NMI_{SM}	NMI_{SI}	ARI_{SM}	ARI_{SI}	AWI_{SM}	AWI_{SI}
Madrid bombing terrorists	6	5	26	0.50	0.44	0.33	0.27	0.83	0.73
Star Wars characters	9	5	35	0.37	0.53	0.15	0.28	0.39	0.94
Australian University Campus	13	6	30	0.47	0.72	0.20	0.46	0.76	0.93
‘Les Miserables’ characters	8	6	33	0.56	0.56	0.46	0.50	0.92	0.83

Weighted networks	# communities								
	Modularity	Infomap	Surprise	NMI_{SM}	NMI_{SI}	ARI_{SM}	ARI_{SI}	AWI_{SM}	AWI_{SI}
Madrid bombing terrorists	8	5	22	0.50	0.67	0.37	0.62	0.65	0.94
Star Wars characters	10	6	52	0.36	0.43	0.18	0.24	0.79	0.79
Australian University Campus	14	6	25	0.47	0.74	0.22	0.46	0.72	0.91
‘Les Miserables’ characters	10	6	32	0.51	0.61	0.37	0.49	0.72	0.74

Table 8: Table comparing the performance of modularity, Infomap and surprise on the bunch of real-world networks considered here. While Infomap is the algorithm producing the smallest number of clusters, surprise is the one producing the largest number of clusters; as a consequence, the three partitions output by our algorithms have an overall small overlap. Interestingly enough, the values of the AWI are quite large: since it just focuses on the percentage of true positives, a good performance under such an index indicates that the tested algorithms ‘agree’ on the nodes to be clustered together: hence, the discrepancy between the ARI and the AWI may be explained by the presence of statistical ‘noise’ (i.e. pairs of nodes on whose classification the algorithms ‘disagree’) around the bulk of nodes to be put together - on which the algorithms ‘agree’.

its translation into a routine require an amount time - not to mention the technical skills required for the task - that is too large if compared with the benefits deriving from its application. For this reason, we are convinced the success of a certain algorithm to be strongly influenced by the availability of a routine, written in a popular programming language such as Python, that allows other researchers to (easily) install and use it. In this sense, the 'SurpriseMeMore' module fills this gap by implementing all the surprise-based methods considered in the present chapter[7].

The 'SurpriseMeMore' library [115] can be easily installed from the Python Package Index (PyPi) by tapping

```
pip install surprisememore
```

in your terminal[8].

3.7.1 Graph instances

As for other network libraries (e.g. `networkx`), the basic types of 'SurpriseMeMore' are graphs: the initialization of a *Directed* or *Undirected* graph instance can be done by providing either an adjacency matrix or an edgelist. An example code for initializing an undirected graph follows:

```
from surprisememore import UndirectedGraph
graph = UndirectedGraph(adjacency=adj_undir)
```

(where *adj_undir* is the adjacency matrix of the considered network). Similarly, the `DirectedGraph` class can be imported, and used, to initialize a directed graph instance.

[7]Although our package is not the only one implementing surprise-based methods for the detection of mesoscale structures (see, for example, [113, 114]), it is the only one implementing the weighted variants of surprise-based methods.

[8]For more details about the supported Python versions and how to deal with potential compatibility errors, the reader is redirected to the Readme file that can be found on the Github page of the package.

These two are the basic types of the 'SurpriseMeMore' module, accepting both binary and weighted adjacency matrices/edgelists[9].

3.7.2 Mesoscale structures detection

In what follows, we briefly present some examples of codes illustrating how to carry out mesoscale structures detection on a given graph. In order to run them, the undirected graph class `UndirectedGraph` must have already been imported.

Modular structures detection

The following snippet of code

```
graph = UndirectedGraph(adjacency=adj_und)
graph.run_discrete_community_detection()
print(graph.solution)
print(graph.surprise)
```

initializes the graph instance `graph`, runs the binary community detection, outputs the partition recovered and its corresponding surprise. The `run_discrete_community_detection` command accepts several, optional arguments as:

- `weighted`: it accepts boolean values, specifying if either the binary or the weighted version of surprise must be optimized;
- `num_sim`: it specifies the number of times the algorithm runs over all the network edges (to search for the configuration optimizing surprise);
- `method`: it defines the surprise optimization strategy. The user can choose between two different methods:
  - `agglomerative`: the algorithm starts with a network of singletons and, then, merges them until a stopping criterion is satisfied;

[9]For more details, the reader is redirected to the Readme file that can be found on the Github page of the package.

    - fixed-clusters: the algorithm starts with a network partitioned into n communities, provided by the users; then, nodes are moved between clusters until a stopping criterion is satisfied. As the number of clusters is a hard constraint, the algorithm will look for the optimum value of surprise compatible with the given number of communities.

  The default value of the argument is agglomerative;

- n_clusters: it indicates the number of communities and has to be specified when the fixed-clusters method is used;
- initial_guess: it provides the initial guess for the algorithm, the available choices for the user being random and common-neighbours.

Interested readers are redirected to the Github page of the 'Surprise-MeMore' module where more information about the methods for modular structures detection is available.

Bimodular structures detection

As in the previous paragraph, let us first introduce a snippet of code that can be used to detect bimodular structures in networks:

```
graph = UndirectedGraph(adjacency=adj_und)
graph.run_discrete_cp_detection()
print(graph.solution)
print(graph.surprise)
```

naturally, the function run_discrete_cp_detection accepts optional arguments as well:

- weighted: it accepts boolean values, specifying if either the binary or the weighted version of surprise must be optimized;
- num_sim: it specifies the number of times the algorithm runs over all the network edges (to search for the configuration optimizing surprise);

- `initial_guess`: it provides the initial guess for the algorithm, the available choices for the user being `random`, `ranked` and `eigenvector`.

Interested readers are redirected to the Github page of the 'Surprise-MeMore' module where more information about the methods for bimodular structures detection is available.

Chapter 4

The weighted Bitcoin Lightning Network

This chapter is devoted to present the results of the paper [116], submitted to Physical Review Applied and dealing with some applications of the methods introduced in the previous two chapters - specifically, to the weighted Bitcoin Lightning Network. The latter, whose purely binary topology has been studied in [55, 49], is an ideal benchmark to test our methods, for a couple of reasons. First, its size grows in time: hence, scalable methods for the definition of null models are required; second, it is a weighted network: hence, can be employed to test both the binary and the weighted versions of surprise introduced in the previous chapter (e.g. to individuate the presence of a core-periphery structure, check if the presence of weights impacts on the latter and how, etc.).

4.1 Introduction

The *Bitcoin Lightning Network* (BLN) represents an attempt to overcome one of the main limitations of the Bitcoin technological design, i.e. *scalability*: at the moment, only a limited amount of transactions per second (whose number is proportional to the size of blocks and their release frequency) can be processed by Bitcoin, a major shortcoming preventing the adoption of this payment system at a global scale - especially when considering that classic payment mechanisms are able to achieve tens of thousands of transactions per second.

Increasing the size of the blocks has been proposed as a solution; implementing this choice, however, would require 1) a larger validation time, 2) a larger storage capability and 3) larger bandwidth costs, hence favoring a more centralized validation process: in fact, fewer entities would become able to validate the new blocks, thus making the system as a whole more prone to faults and attacks. Developers have tried to break the trade-off between block size and centralization by proposing to process transactions off-chain, i.e. by means of a 'Layer 2' protocol that can operate on top of blockchain-based cryptocurrencies such as Bitcoin: such a protocol, known with the name of Bitcoin Lightning Network, works by creating payment channels across which any two users can exchange money without having the data related to their transactions burdening the entire blockchain.

The BLN has recently raised a lot of interest: Lee et al. [117] showed that it is characterized by a scale-free topology; Lin et al. [55] and Martinazzi et al. [118] analyzed the evolution of the BLN topology and found it to have become increasingly centralized at different levels; Seres et al. [119] argued that the BLN structure can be ameliorated to improve its security; the authors of [120] and [121] showed that the current BLN can be prone to channel exhaustion or attacks aimed at isolating nodes, thus compromising their reachability, the payment success ratio, etc. Mizrahi et al. [122] analyzed the robustness of the BLN against three different types of attacks: locking channels, disconnecting pairs of nodes and isolating hubs; although their results indicate that the BLN can be disrupted at a relatively low cost, Conoscenti et al. [123] suggested that the BLN is still resilient against the removal of nodes that do not have a significant influence on the probability of success of a payment.

Most of the aforementioned contributions, however, just focused on the analysis of the BLN binary structure, leaving its weighted counterpart largely unexplored. With the present paper we aim at filling this gap, by studying the weighted properties of the BLN daily snapshot representation, at the micro-, meso- and macro-scale, across a period of 18 months, i.e. from 12th January 2018 to 17th July 2019.

4.2 Data

As it was the authors of [55] explained, payments in the BLN are *source-routed* and *onion-routed*: hence, in order to pre-compute the entire payment route, the sender must have a reasonably up-to-date view of the network topology. Nodes in the BLN regularly broadcast information about the channels they participate in: such a mechanism, called *gossip*, allows other nodes to keep their view of the network topology up-to-date.

The BLN topology can be visualized by means of the the so-called *routing table*. For this paper, we took a snapshot of the routing table every 15 minutes, between January 12th 2018, at blockheight 503.816, to July 17th 2019, at blockheight 585.844: these snapshots were, then, aggregated into *timespans*, each timespan representing a constant state of a channel from its start to its end; for the present analysis, we considered the *daily snapshot* representation of the BLN, including all channels that were found to be active during that day. Importantly, here we do not rest upon estimates of the number of daily blocks - obtainable by considering that the time between the appearance of two subsequent blocks, in the blockchain, is Poisson distributed with an expected value of 10 minutes - but on the exact time our channels have been opened: since every channel consists of an unspent transaction output on the blockchain, we can determine the size of a channel, its opening and closing time within minutes.

4.3 Methods

4.3.1 Notation

On a generic, daily snapshot t, the BLN can be described as a weighted, undirected network with total number of nodes $N^{(t)}$ and represented by an $N^{(t)} \times N^{(t)}$ symmetric matrix $\mathbf{W}^{(t)}$ whose generic entry $w_{ij}^{(t)}$ indicates

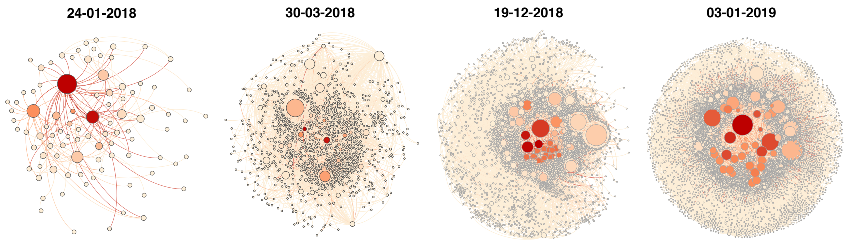


Figure 20: Pictorial representation of the four snapshots of the BLN, corresponding to the days 24-01-2018, 30-03-2018, 19-12-2018, 01-03-2019 and whose LCC is characterized by a number of nodes amounting at 100, 1.000, 3.000, 5.000, respectively. The size of each node is proportional to its degree (i.e. the bigger the node, the larger its degree) while the color of each node is proportional to its strength (i.e. the darker the node, the larger its strength).

the total amount of money exchanged between i and j, across all channels established by them, during the snapshot t [124, 125]. Consistently, the generic entry of the BLN binary adjacency matrix $\mathbf{A}^{(t)}$ reads $a_{ij}^{(t)} = 1$ if $w_{ij}^{(t)} > 0$ and $a_{ij}^{(t)} = 0$ otherwise: the presence of a link between any two nodes i and j, i.e. $a_{ij}^{(t)} = 1$, indicates that one or more payment channels have been opened, between the same nodes, during the snapshot t. As a last remark, we will focus on the largest connected component (LCC) of the BLN, throughout its entire history - the percentage of nodes belonging to it being steadily above 90%. For the sake of illustration, we will plot our results for four snapshots, i.e. the ones whose LCC is characterized by a number of nodes amounting at 100, 1.000, 3.000, 5.000 and corresponding to the days 24-01-2018, 30-03-2018, 19-12-2018 and 01-03-2019, respectively - see fig. 20.

4.3.2 Degree and strength distributions

The total number of channels (i.e. *links*) that have been opened during the snapshot t is provided by $L^{(t)} = \sum_{i=1}^{N^{(t)}} \sum_{j=i+1}^{N^{(t)}} a_{ij}^{(t)}$; on the other hand, the total number of channels node i participates in coincides with its *degree*, i.e. $k_i^{(t)} = \sum_{j(\neq i)=1}^{N^{(t)}} a_{ij}^{(t)}$. The weighted counterparts of the notions above coincide with the total weight of the network, i.e. $W^{(t)} = \sum_{i=1}^{N^{(t)}} \sum_{j=i+1}^{N^{(t)}} w_{ij}^{(t)}$, and with the total amount of money exchanged by node i, i.e. $s_i^{(t)} = \sum_{j(\neq i)=1}^{N^{(t)}} w_{ij}^{(t)}$, a quantity often referred to as node *strength* or node *capacity*.

While inspecting the functional form of the degree and strength distributions may reveal the presence of hubs, i.e. 'large', single nodes, when dealing with cryptocurrencies it is of interest making a step further and inspecting the presence of 'large subgraphs' of nodes. The meaning of this sentence can be made more precise upon considering the metric designed by Srinivasan et al. [126] to measure the number of addresses required (to collude) for gathering over the 51% of the overall mining power and named *Nakamoto index* - more explicitly, a high Nakamoto coefficient indicates that many miners, or mining pools, need to combine their power to reach the 51% threshold needed to take over the blockchain. Here, we adapt it to quantify a 'topological' kind of majority, by defining

$$N_k = \min\{i \in [1 \dots N] : \sum_i^N f_i \geq 0.51\} \tag{4.1}$$

where $f_i = k_i/2L$ and

$$N_s = \min\{i \in [1 \dots N] : \sum_i^N f_i \geq 0.51\} \tag{4.2}$$

where $f_i = s_i/2W$: the first variant of the Nakamoto index can be calculated by starting from the (node with) largest degree and add them up until the condition above is satisfied; analogously, for the second variant.

4.3.3 Assortativity and hierarchy

In order to gain insight into the higher-order structure of the BLN, in what follows we will consider the quantities known as *average nearest neighbors degree*, defined as

$$\mathrm{ANND}_i = \frac{\sum_{j(\neq i)=1}^{N} a_{ij} k_j}{k_i}, \quad \forall\, i \tag{4.3}$$

and *average nearest neighbors strength*, defined as

$$\mathrm{ANNS}_i = \frac{\sum_{j(\neq i)=1}^{N} a_{ij} s_j}{k_i}, \quad \forall\, i; \tag{4.4}$$

while plotting ANND_i versus k_i reveals the (either positive or negative) assortative character of a network, i.e. the presence of (either positive or negative) correlations between degrees, plotting ANNS_i versus k_i reveals the presence of (either positive or negative) correlations between degrees and strengths.

On the other hand, the 'cohesiveness' of the neighborhood of each node can be inspected by calculating the *binary clustering coefficient*

$$\mathrm{BCC}_i = \frac{\sum_{j(\neq i)=1}^{N} \sum_{k(\neq i,j)=1}^{N} a_{ij} a_{jk} a_{ki}}{k_i(k_i - 1)}, \quad \forall\, i \tag{4.5}$$

defined as the percentage of triangles established by any two neighbors of each node and the node itself; its weighted counterpart reads

$$\mathrm{WCC}_i = \frac{\sum_{j(\neq i)=1}^{N} \sum_{k(\neq i,j)=1}^{N} w_{ij} w_{jk} w_{ki}}{k_i(k_i - 1)}, \quad \forall\, i \tag{4.6}$$

and is intended to assign a 'weight' to each triangle counted by the BCC, by weighing the connections shaping it. Plotting BCC_i versus k_i reveals the (possibly) hierarchical character of a network, i.e. its organization in sub-modules; plotting WCC_i versus k_i, instead, provides a hint about the magnitude of the nodes inter-connections as a function of the nodes connectivity.

4.3.4 Disparity

The *disparity index* is defined as

$$Y_i = \sum_{j(\neq i)=1}^{N} \left[\frac{w_{ij}}{s_i}\right]^2 = \frac{\sum_{j(\neq i)=1}^{N} w_{ij}^2}{s_i^2} = \frac{\sum_{j(\neq i)=1}^{N} w_{ij}^2}{\left[\sum_{j(\neq i)=1}^{N} w_{ij}\right]^2}, \quad \forall\, i \tag{4.7}$$

and quantifies the (un)evenness of the distribution of the weights 'constituting' the i-th strength over the k_i links characterizing the connectivity of node i. More specifically, the disparity index of node i reads $Y_i = 1/k_i$ in case weights are equally distributed among the connections established by it, i.e. in case $w_{ij} = a_{ij}s_i/k_i$, $\forall\, j$, any larger value signaling an excess concentration of weight in one or more links.

4.3.5 Centrality

Any index measuring the centrality of a node aims at quantifying its importance in the network, according to some, specific, topological criterion [127, 128, 4, 129]. While the efforts of researchers have mainly focused on the definition of binary centrality measures, relatively little work has been done on their weighted counterparts. In order to fill this gap, in what follows, we will consider possible extensions of the centrality measures employed in [55], i.e. the *degree*, *closeness*, *betweenness* and *eigenvector* centrality:

Weighted degree centrality

The degree centrality [4, 129] of node i coincides with its degree, normalized by the maximum attainable value, i.e. $\text{DC}_i = k_i/(N-1)$: the strength centrality of node i generalizes it by simply replacing the total number of 'node-specific' connections with the total 'node-specific' weight. In what follows we will consider the (simpler) definition

$$\text{WDC}_i = s_i, \quad \forall\, i \tag{4.8}$$

formalizing that the most central node, according to the strength variant, is the one characterized by the largest percentage of weight 'embodied' by (the totality of) its connections.

Weighted closeness centrality

The closeness centrality [4, 129] of node i is defined as $\mathrm{CC}_i = (N - 1)/\sum_{j(\neq i)=1}^{N} d_{ij}$ where d_{ij} is the topological distance between nodes i and j, i.e. the length of any shortest path connecting them. The definition of weighted closeness centrality of node i is based on the redefinition of shortest path length which, in turn, rests upon the redefinition of *weighted distance* between any two nodes, i.e. $d_{ij}^{(w)}$. Possible variants of the latter one read $d_{ij}^{(w)} = \min\{w_{ih} + \dots + w_{hj}\}$ and $d_{ij}^{(w)} = \min\left\{\frac{1}{w_{ih}} + \dots + \frac{1}{w_{hj}}\right\}$ where h indexes the intermediary vertices lying on the path between i and j, $w_{ih} \dots w_{hj}$ are the weights of the corresponding edges and the extremum is taken over all paths between i and j. Naturally, the meaning changes along with the chosen definition: while the first one describes any two nodes as 'closer', the smaller the weights of the intermediate connections, the opposite is true when the second one is considered. Hereby, we opt for the definition of weighted closeness centrality reading

$$\mathrm{WCC}_i = \frac{N-1}{\sum_{j(\neq i)=1}^{N} d_{ij}^{(w)}}, \quad \forall\, i \tag{4.9}$$

with $d_{ij}^{(w)} = \min\left\{\frac{1}{w_{ih}} + \dots + \frac{1}{w_{hj}}\right\}$ - a choice implying that once the path connecting nodes i and j has been individuated, the WCC is nothing else that the harmonic mean of the weights constituting it.

Weighted betweeness centrality

The betweenness centrality [4, 130, 131, 132] of node i is given by $\mathrm{BC}_i = \sum_{s(\neq i)=1}^{N} \sum_{t(\neq i,s)=1}^{N} \sigma_{st}(i)/\sigma_{st}$ where σ_{st} is the total number of shortest paths between node s and t and $\sigma_{st}(i)$ is the number of shortest paths between nodes s and t that pass through node i. The weighted counterpart of it can be defined as

$$\mathrm{WBC}_i = \sum_{s(\neq i)=1}^{N} \sum_{t(\neq i,s)=1}^{N} \frac{\sigma_{st}^{(w)}(i)}{\sigma_{st}^{(w)}}, \quad \forall\, i \tag{4.10}$$

where, now, $\sigma_{st}^{(w)}$ is the total number of weighted shortest paths between nodes s and t and $\sigma_{st}^{(w)}(i)$ is the number of weighted shortest paths between nodes s and t that pass through node i.

Weighted eigenvector centrality

The eigenvector centrality [4, 133, 132] of node i is defined as the i-th element of the eigenvector corresponding to the largest eigenvalue of the binary adjacency matrix - whose existence is guaranteed in case the Perron-Frobenius theorem holds true. According to the definition above, a node with large eigenvector centrality is connected to other 'well connected' nodes. Such a definition can be extended by considering the WEC$_i$, defined as the i-th element of the eigenvector corresponding to the largest eigenvalue of the weighted adjacency matrix.

The Gini coefficient

The Gini coefficient, defined as

$$G_c = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}|c_i - c_j|}{2N\sum_{i=1}^{N}c_i} \tag{4.11}$$

has been introduced to quantify 'inequality' in wealth distributions [134, 135] and ranges between 0 and 1, a larger Gini coefficient indicating a larger '(un)evenness' of the income distribution. Hereby, we apply it to the several definitions of centrality provided above[1].

4.3.6 Small-world -ness

The study of the BLN centralization can be approached from a slightly different perspective by asking if the BLN is (increasingly) becoming a small-world system [136, 137, 138]. The usual way of proceeding to answer such a question prescribes to check if

[1]As a general comment, we would like to stress that a non-normalized centrality measure cannot be employed to compare nodes, across different configurations, in a fully consistent way. However, if our only interest is that of quantifying the (un)evenness of the distribution of our centrality measures, the absence of a normalization term does not make any difference: in fact, the Gini coefficient is not affected by it.

$$\overline{d} = \frac{\sum_{i=1}^{N} \sum_{j(\neq i)=1}^{N} d_{ij}}{N(N-1)} \sim \ln N \tag{4.12}$$

i.e. if the average path length grows logarithmically with the number of nodes and if the average clustering coefficient $\overline{\mathrm{BCC}}_i = \sum_{i=1}^{N} \mathrm{BCC}_i / N$ is larger than the one predicted by an *Undirected Random Graph Model* (URGM) tuned to reproduce the empirical density of links.

Recently, however, it has been argued that the same question can be answered by considering the quantity named *global efficiency*, defined as

$$E_g = \frac{\sum_{i=1}^{N} \sum_{j(\neq i)=1}^{N} d_{ij}^{-1}}{N(N-1)} \tag{4.13}$$

and understood as an indicator of the 'traffic capacity' of a network and, quite remarkably, not affected by the analytical problems suffered by the average path length [137] - potentially diverging due to the presence of couples of nodes belonging to disconnected components. Latora et al. [137] have also defined the *local efficiency*

$$E_l = \frac{1}{N} \sum_{i=1}^{N} E(\mathbf{G}_i), \tag{4.14}$$

a quantity that can be evaluated by, first, calculating the efficiency of the subgraph induced by the nearest neighbors of each node, upon removing it, and, then, averaging such numbers. Latora et al. [137] have argued that while E_g plays a role analogous to the inverse of the average path length, E_l plays a role analogous to the average clustering coefficient: hence, small-world networks should have both a large E_g and a large E_l, i.e. should be very efficient in allowing nodes to communicate in both a global and a local fashion.

4.3.7 Core-periphery detection

As it has emerged quite clearly from the binary analysis of the BLN, just inspecting the evolution of centrality measures can return a too simplistic picture of the network under consideration. For this reason, we have checked for the presence of mesoscopic 'centralized' structures such as the *core-periphery* one, composed by a densely-connected subgraph of nodes surrounded by a periphery of loosely-connected vertices. In order to do so, we have implemented the approach discussed in the previous chapter and prescribing to minimize the weighted, bimodular surprise $\mathscr{W}_{/\!/}$, whose definition is recalled here[2]:

$$\begin{aligned}\mathscr{W}_{/\!/} &= \sum_{w_\bullet \geq w_\bullet^*} \sum_{w_\circ \geq w_\circ^*} f(w_\bullet, w_\circ) \\ &= \sum_{w_\bullet \geq w_\bullet^*} \sum_{w_\circ \geq w_\circ^*} \frac{\binom{V_\bullet + w_\bullet - 1}{w_\bullet}\binom{V_\circ + w_\circ - 1}{w_\circ}\binom{V-(V_\bullet+V_\circ)+W-(w_\bullet+w_\circ)-1}{W-(w_\bullet+w_\circ)}}{\binom{V+W-1}{W}} \\ &= \sum_{w_\bullet \geq w_\bullet^*} \sum_{w_\circ \geq w_\circ^*} \frac{\left(\!\binom{V_\bullet}{w_\bullet}\!\right)\left(\!\binom{V_\circ}{w_\circ}\!\right)\left(\!\binom{V-V_\bullet-V_\circ}{W-w_\bullet-w_\circ}\!\right)}{\left(\!\binom{V}{W}\!\right)}; \end{aligned} \tag{4.15}$$

the expression above has been written by employing the multiset notation, according to which $\left(\!\binom{V_\bullet}{w_\bullet}\!\right) = \binom{V_\bullet + w_\bullet - 1}{w_\bullet}$ (see also table 7 in the previous chapter).

4.3.8 Benchmarking the observations

Along the guidelines of the analysis carried out in [55], in what follows we benchmark our observations by employing the recently-proposed null model called CReM$_\text{A}$ - an acronym standing for *Conditional Reconstruction Model A* [62, 31] - that allows binary and weighted constraints to be defined in a disentangled fashion. From a purely theoretical point of view, it is defined by the maximization of the *conditional Shannon entropy*

$$S(\mathscr{W}|\mathscr{A}) = -\sum_{\mathbf{A}\in\mathbb{A}} P(\mathbf{A}) \int_{\mathbb{W}_\mathbf{A}} Q(\mathbf{W}|\mathbf{A}) \ln Q(\mathbf{W}|\mathbf{A}) d\mathbf{W} \tag{4.16}$$

[2]Briefly, $V = N(N-1)/2$ is the total number of node pairs, $W = \sum_{i=1}^N \sum_{j=i+1}^N w_{ij}$ is the total weight of the network, $V_\bullet$ is the number of node pairs in the core portion of the network, $V_\circ$ is the number of node pairs in the periphery portion of the network, $w_\bullet^*$ is the observed number of core links and $w_\circ^*$ is the observed number of periphery links.

constrained to reproduce the strengths $\{s_i\}_{i=1}^N$; the (conditional) weighted distribution output by such an optimization procedure reads

$$Q(\mathbf{W}|\mathbf{A}) = \frac{e^{-H(\mathbf{W})}}{Z_\mathbf{A}} = \prod_{i=1}^{N}\prod_{j=i+1}^{N} q_{ij}(w_{ij}|a_{ij}) \\ = \prod_{i=1}^{N}\prod_{j=i+1}^{N} (\beta_i + \beta_j)^{a_{ij}} e^{-(\beta_i+\beta_j)w_{ij}}; \qquad (4.17)$$

notice the conditional character of the distribution above, embodied by the term a_{ij} at the exponent[3]. The vector of parameters $\{\beta_i\}_{i=1}^N$ defining the distribution above can be estimated via a (generalized) *likelihood maximization* procedure [62] that leads to the system of N equations

$$s_i = \sum_{\substack{j=1 \\ (j\neq i)}}^{N} \langle w_{ij}\rangle = \sum_{\substack{j=1 \\ (j\neq i)}}^{N} \frac{p_{ij}}{\beta_i + \beta_j}, \quad \forall\, i; \qquad (4.18)$$

the coefficients $\{p_{ij}\}_{i,j=1}^N$, instead, are treated as 'prior information' and, as such, left 'untouched' by the estimation procedure above. In a sense, we are free to combine the (conditional) weighted distribution above with the purely binary probability mass function 'best' encoding the available information about the network structure.

In what follows, we have considered

- the binary probability mass function defining the *Undirected Binary Configuration Model* (UBCM) and following from the maximization of the traditional Shannon entropy $S = -\sum_\mathbf{A} P(\mathbf{A}) \ln P(\mathbf{A})$ constrained to reproduce the degrees $\{k_i\}_{i=1}^N$: the UBCM captures the idea that the probability for any two nodes to establish a connection (solely) depends on their degrees and can be fully determined by solving the N equations

[3]As a simple consistency check, the probability that $w_{ij} = 0$ in case there is no link is $q(w_{ij} = 0|a_{ij} = 0) = 1$ as it should be.

$$k_i = \sum_{\substack{j=1\\(j\neq i)}}^{N} p_{ij}^{\text{UBCM}} = \sum_{\substack{j=1\\(j\neq i)}}^{N} \frac{x_i x_j}{1+x_i x_j}, \quad \forall\, i; \tag{4.19}$$

- the deterministic recipe $p_{ij} \equiv a_{ij}, \forall\, i < j$, accounting for the case in which the prior knowledge concerns the entire network topological structure, now treated as given.

In the first case, the generic set of coefficients $\{p_{ij}\}_{i,j=1}^{N}$ is instantiated upon identifying $p_{ij} \equiv p_{ij}^{\text{UBCM}}$, $\forall\, i < j$; in the second one, instead, the identification simply reads $p_{ij} \equiv a_{ij}$, $\forall\, i < j$; in both cases, the resolution of the related system of equations, carried out via the 'NEMTROPY' package, leads us to numerically determine the corresponding vector of parameters $\{\beta_i\}_{i=1}^{N}$.

Benchmarking a set of observations ultimately boils down at verifying their 'compatibility' with the predictions output by a chosen null model, by testing their statistical significance against the null model itself. To this aim, one can proceed as follows: 1) sampling the ensemble induced by the chosen null model, by generating a 'sufficiently large' number of configurations (in all our experiments, 100); 2) calculating the value of any quantity of interest over each configuration; 3) deriving the corresponding ensemble CDF. At this point, a p-value remains naturally defined; in what follows, we will employ it to carry out one-tailed tests. Whenever tests of this kind are considered, one may be interested in calculating either the (ensemble) probability $Q(X \geq X^*)$ of observing a value, for the quantity of interest X, that is *larger* than the empirical one, X^*, or the (ensemble) probability $Q(X \leq X^*)$ of observing a value, for the quantity of interest X, that is *smaller* than it; in both cases, if such a probability is found to be smaller than a given threshold, the quantity is deemed as statistically-significant, hence incompatible with the description provided by the chosen null model - which (significantly) underestimates or overestimates it, respectively.

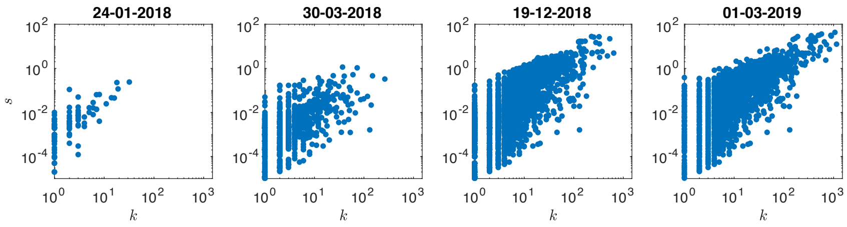


Figure 21: Scattering the strength sequence versus the degree sequence reveals the presence of a positive correlation between the two sets of quantities: the Pearson coefficient describing it, on our usual four snapshots, amounts at $r = 0.84, 0.42, 0.66, 0.80$, respectively.

4.4 Results

4.4.1 Degree and strength distributions

The four snapshots depicted in fig. 20 reveal the presence of a large heterogeneity, with nodes having a large degree/strength co-existing with nodes having a small degree/strength; moreover, while nodes with a large degree also have a large strength (i.e. larger nodes are also darker), small, dark nodes can be observed as well: in other words, an overall positive correlation between degrees and strengths co-exists with a large variability of the strength values - especially for what concerns the nodes with a small connectivity (see fig. 21).

As a first empirical analysis, we have inspected the functional form of the degree distribution for our usual four snapshots, i.e. the days 24-01-2018, 30-03-2018, 19-12-2018 and 01-03-2019; to this aim, we have plotted the cumulative density function (CDF), defined as $\text{CDF}(k) = \sum_{h \geq k} f(h)$ where $f(h)$ is the fraction of nodes whose degree is h. As shown in fig. 22, the support of the degree distribution becomes broader as the BLN evolves; moreover, running the code released by Clauset et al. [139] to fit the functional form $\text{PDF}(k) = (\alpha - 1)k_{min}^{\alpha-1}k^{-\alpha}$ on the data returns the values $\alpha = 1.9, 2.0, 2.1, 2.2$ and $k_{min} = 1, 3, 14, 26$ while the Kolmogorov-Smirnov test returns the p-values $p = 0.02, 0.03, 0.04, 0.5$. Hence, the null

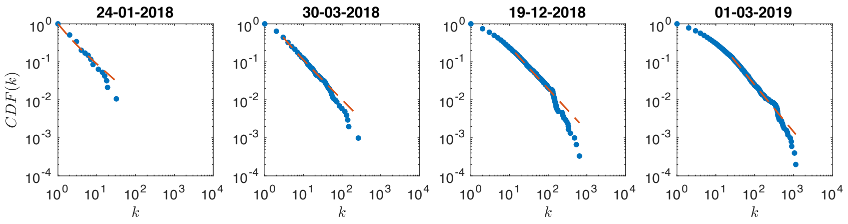


Figure 22: Cumulative density function of the degrees, for our usual four snapshots. The support of the distribution has become broader as the BLN has evolved. Fitting a power-law $\mathrm{PDF}(k) = (\alpha - 1)k_{min}^{\alpha-1}k^{-\alpha}$ on the data (naturally, for $k \geq k_{min}$), by running the code released by Clauset et al. [139], returns values amounting at $\alpha = 1.9, 2.0, 2.1, 2.2$ and $k_{min} = 1, 3, 14, 26$ while the Kolmogorov-Smirnov test returns the p-values $p = 0.02, 0.03, 0.04, 0.5$. Hence, the null hypothesis that the degrees are distributed according to a power-law is never rejected, at the 1% significance level - while it is, for the first three snapshots, at the 5% significance level. Overall, the null hypothesis that the degrees are distributed according to a power-law is not rejected for the 85% of the total number of snapshots, at the 1% significance level, and for the 71% of the total number of snapshots, at the 5% significance level.

hypothesis that the degrees are distributed according to a power-law is never rejected, at the 1% significance level - while it is, for the first three snapshots, at the 5% significance level. Overall, the null hypothesis that the degrees are distributed according to a power law is not rejected for the 85% of the total number of snapshots, at the 1% significance level, and for the 71% of the total number of snapshots, at the 5% significance level.

As a second empirical analysis, we have calculated the evolution of the CDF of the weights, defined as $\mathrm{CDF}(w) = \sum_{v \geq w} f(v)$. Analogously to what has been observed for the degrees, even the support of the weight distribution has broadened throughout the entire BLN history (see fig. 23), although to a lesser extent. Fitting a log-normal distribution, whose functional form reads $\mathrm{PDF}(w) = (w\sigma\sqrt{2\pi})^{-1}e^{-\frac{(\ln w - \mu)^2}{2\sigma^2}}$, on the data reveals that, at both the 1% and the 5% significance levels, the Kolmogorov-

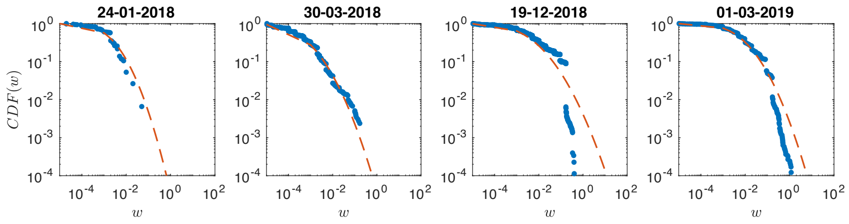


Figure 23: Cumulative density function of the weights, for our usual four snapshots. The support of the distribution has become slightly broader as the BLN has evolved. Fitting a log-normal distribution $\text{PDF}(w) = (w\sigma\sqrt{2\pi})^{-1}e^{-\frac{(\ln w-\mu)^2}{2\sigma^2}}$ on the data reveals that, at both the 1% and the 5% significance levels, the Kolmogorov-Smirnov test does not reject the hypothesis that weights are log-normally distributed when $N < 94$ (i.e. from the fourth day to the twelfth day); for our four snapshots, instead, the hypothesis is rejected - notice that day 24-01-2018 is the thirteenth.

Smirnov test does not reject the hypothesis that weights are log-normally distributed when $N < 94$ (i.e. from the fourth day to the twelfth day): for our four snapshots, instead, the hypothesis is rejected - notice that day 24-01-2018 is the thirteenth.

As a third empirical analysis, we have considered the evolution of the CDF of the strengths, defined as $\text{CDF}(s) = \sum_{t\geq s} f(s)$. The support of the distribution is enlarged of a few orders of magnitude during the BLN history (see fig. 24). Analogously to the case of the weights, we have fitted a log-normal distribution, whose functional form reads $\text{PDF}(s) = (s\sigma\sqrt{2\pi})^{-1}e^{-\frac{(\ln s-\mu)^2}{2\sigma^2}}$, on the data: while the the Kolmogorov-Smirnov test returns the p-values $p = 0.061, 0.006, 4.4\cdot 10^{-7}$ and $1.6\cdot 10^{-7}$, hence does not reject the hypothesis that strengths are log-normally distributed, at both significance levels, for the first snapshot considered here, it does so for the other three ones - an evidence seemingly indicating that, quite early in its history, the BLN has started deviating more and more from the picture provided by the distribution tested here. Overall, the null hypothesis that the strengths are distributed according to a log-

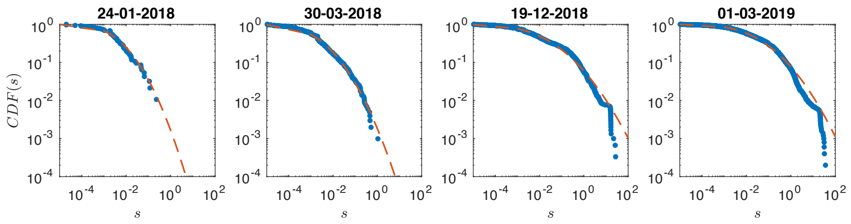


Figure 24: Cumulative density function of the strengths, for our usual four snapshots. The support of the distribution is enlarged of a few orders of magnitude during the BLN history. A log-normal distribution $\text{PDF}(s) = (s\sigma\sqrt{2\pi})^{-1}e^{-\frac{(\ln s-\mu)^2}{2\sigma^2}}$, fitted on the data, lets the Kolmogorov-Smirnov test return the p-values $p = 0.061, 0.006, 4.4 \cdot 10^{-7}$ and $1.6 \cdot 10^{-7}$. Hence, the null hypothesis that strengths are log-normally distributed is not rejected for the first snapshot while it is for the last three ones - at both significance levels. Overall, the null hypothesis that the strengths are distributed according to a log-normal is not rejected for the 16% of the total number of snapshots, at the 1% significance level, and for the 5% of the total number of snapshots, at the 5% significance level.

normal is not rejected for the 16% of the total number of snapshots, at the 1% significance level, and for the 5% of the total number of snapshots, at the 5% significance level.

The picture provided by the three distributions above can be complemented by the information provided by the Nakamoto index (see fig. 25). As its evolution clearly shows, the percentage of nodes 'providing' the 51% of the total number of links/the total weight progressively reduces, as the BLN size enlarges: in particular, the total weight seems to be distributed less evenly than the total number of connections - as fewer nodes are needed to embody the (same) required percentage. This seems to confirm the appearance of nodes constituting the aforementioned 'topological' majority.

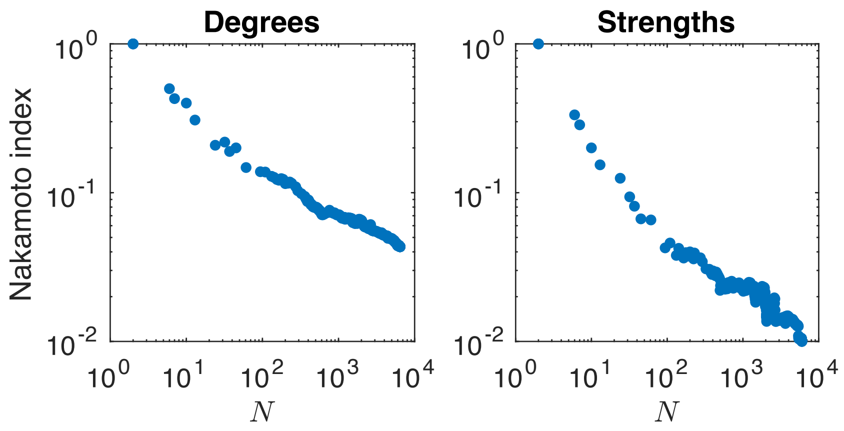


Figure 25: Evolution of the Nakamoto index for the degrees and the strengths, plotted versus the total number of nodes: as the size of the system enlarges, the percentage of nodes 'providing' the 51% of the total number of links/the total weight progressively reduces, an evidence pointing out that nodes embodying a 'topological' kind of majority indeed appear. Moreover, the total weight seems to be distributed less evenly than the total number of connections.

4.4.2 Assortativity and hierarchy

Plotting the values of the average nearest neighbors degree versus the degrees reveals the disassortative character of the BLN, i.e. the presence of negative correlations between the degrees: in other words, nodes with a large degree are (preferentially) connected to nodes with a small degree and viceversa. To be noticed that the UBCM-induced CReM$_\text{A}$ model successfully captures such a trend, indicating that the information encoded into the degree sequence, leading to

$$\langle \text{ANND}_i \rangle \simeq \frac{\sum_{j(\neq i)=1}^{N} \langle a_{ij} \rangle \langle k_j \rangle}{\langle k_i \rangle} = \frac{\sum_{j(\neq i)=1}^{N} p_{ij} k_j}{k_i}, \quad \forall\, i \tag{4.20}$$

(with $p_{ij} \equiv p_{ij}^{\text{UBCM}}$, $\forall\, i < j$ and where the symbol $\simeq$ indicates that we have approximated the expected value of a ratio as the ratio of the ex-

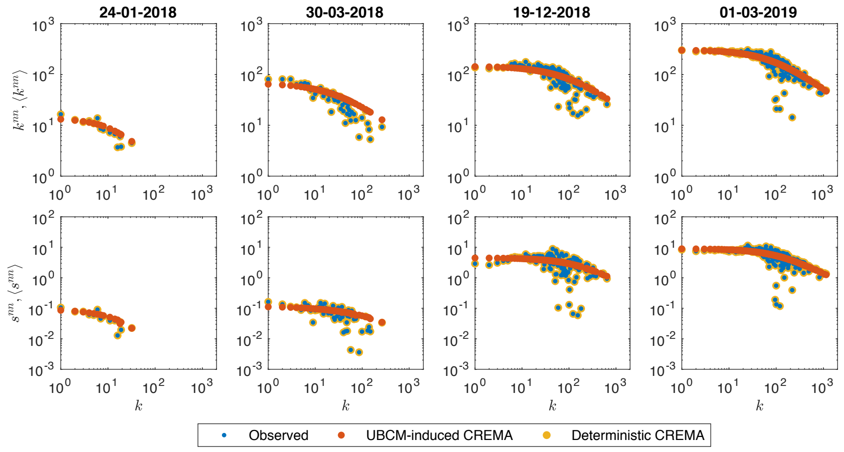


Figure 26: ANND$_i$, $\langle$ANND$_i\rangle$ values scattered versus k_i (upper panels) and ANNS$_i$, $\langle$ANNS$_i\rangle$ values scattered versus k_i (bottom panels) for our usual four snapshots (all trends are averaged over the classes of nodes with the same degree). Both trends clearly signal a disassortative behavior, i.e. nodes with a large degree are (preferentially) connected to nodes with a small degree/small strength and viceversa. While the UBCM-induced CReM$_\text{A}$ model successfully captures such a disassortative trend, the deterministic CReM$_\text{A}$ model reproduces both the ANND and the ANNS values exactly.

pected values) is enough to account for the correlations between the degrees as well. An analogous decreasing trend characterizes the values of the average nearest neighbors strength when plotted versus the degrees, i.e. nodes with a large degree are (preferentially) connected to nodes with a small strength and viceversa; as for its binary counterpart, the UBCM-induced CReM$_\text{A}$ model successfully reproduces the empirical ANNS values, indicating that the information encoded into the degree and the strength sequences, leading to

$$\langle \text{ANNS}_i \rangle \simeq \frac{\sum_{j(\neq i)=1}^{N} \langle a_{ij} \rangle \langle s_j \rangle}{\langle k_i \rangle} = \frac{\sum_{j(\neq i)=1}^{N} p_{ij} s_j}{k_i}, \quad \forall\, i \tag{4.21}$$

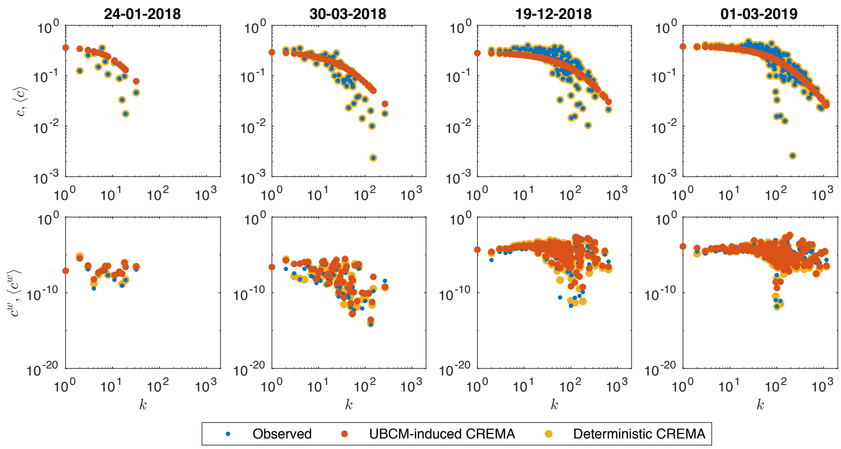


Figure 27: BCC_i, $\langle \text{BCC}_i \rangle$ values scattered versus k_i (upper panels) and WCC_i, $\langle \text{WCC}_i \rangle$ values scattered versus k_i (bottom panels) for our usual four snapshots (all trends are averaged over the classes of nodes with the same degree). While the trend of the BCC clearly signals a hierarchical behavior, i.e. the tendency of nodes with a larger degree to participate into a smaller number of connected triples than nodes with a smaller degree and viceversa, this does not seem to be the case for the WCC values when plotted versus the degrees. While the UBCM-induced CReM_A model successfully captures both trends, the deterministic CReM_A model reproduces only the BCC values exactly.

with $p_{ij} \equiv p_{ij}^{\text{UBCM}}$, $\forall\, i < j$ successfully accounts for the correlations between the degrees and the strengths as well (see fig. 26).

On the other hand, plotting the values of the clustering coefficient versus the degrees reveals the hierarchical character of the BLN: nodes with a larger degree tend to participate into a smaller number of connected triples than nodes with a smaller degree and viceversa; the UBCM-induced CReM_A model, leading to

$$\langle \text{BCC}_i \rangle \simeq \frac{\sum_{j(\neq i)=1}^{N} \sum_{k(\neq i,j)=1}^{N} \langle a_{ij} \rangle \langle a_{jk} \rangle \langle a_{ki} \rangle}{\sum_{j(\neq i)=1}^{N} \sum_{k(\neq i,j)=1}^{N} \langle a_{ij} \rangle \langle a_{ik} \rangle} = \frac{\sum_{j(\neq i)=1}^{N} \sum_{k(\neq i,j)=1}^{N} p_{ij} p_{jk} p_{ki}}{\sum_{j(\neq i)=1}^{N} \sum_{k(\neq i,j)=1}^{N} p_{ij} p_{ik}}, \quad \forall\, i \tag{4.22}$$

with $p_{ij} \equiv p_{ij}^{\text{UBCM}}$, $\forall\, i < j$ is able to capture such a trend as well. The same decreasing trend, instead, does not characterize the values of the weighted clustering coefficient when plotted versus the degrees which, instead, appears as rather flat; interestingly, this is no longer true when the weighted clustering coefficient values are plotted versus the strengths: in this case, a clear rising trend is visible, signalling that nodes with a larger strength tend to participate into 'heavier' connected triples of nodes. Again, the UBCM-induced CReM$_\text{A}$ model, predicting

$$\langle \text{WCC}_i \rangle \simeq \frac{\sum_{j(\neq i)=1}^{N} \sum_{k(\neq i,j)=1}^{N} \langle w_{ij} \rangle \langle w_{jk} \rangle \langle w_{ki} \rangle}{\sum_{j(\neq i)=1}^{N} \sum_{k(\neq i,j)=1}^{N} \langle a_{ij} \rangle \langle a_{ik} \rangle} = \frac{\sum_{j(\neq i)=1}^{N} \sum_{k(\neq i,j)=1}^{N} \langle w_{ij} \rangle \langle w_{jk} \rangle \langle w_{ki} \rangle}{\sum_{j(\neq i)=1}^{N} \sum_{k(\neq i,j)=1}^{N} p_{ij} p_{ik}}, \quad \forall\, i \tag{4.23}$$

with $p_{ij} \equiv p_{ij}^{\text{UBCM}}$, $\forall\, i < j$ successfully reproduces the empirical WCC values, indicating that the information encoded into the degree and the strength sequences successfully accounts for the behavior of third-order properties as well (see fig. 27).

4.4.3 Disparity

As anticipated in the paragraph introducing such a quantity, the disparity index of node i reads $Y_i = 1/k_i$ in case weights are equally distributed among the neighbors of node i. Figure 28 shows the scatter plot of Y_i as a function of k_i (since it is plotted in a log-log scale, the function $y = -x$ becomes the trend signalling that weights are uniformly distributed among the neighbors of each node): generally speaking, many values lie above

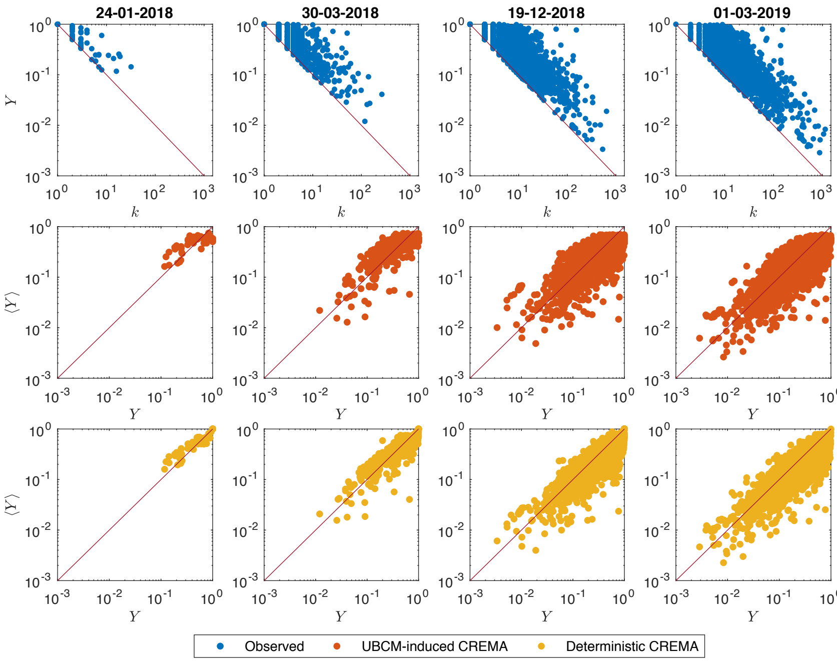


Figure 28: Upper panels: empirical disparity values scattered versus the degrees, for our usual four snapshots. As the plots reveal, the vast majority of strength values is not evenly distributed across the connections characterizing each node, i.e. $Y_i > 1/k_i$ for the vast majority of nodes. Middle panels: expected disparity values output by the UBCM-induced CReM$_\text{A}$ model scattered versus the empirical disparity values. Bottom panels: expected disparity values output by the deterministic CReM$_\text{A}$ model scattered versus the empirical disparity values. The empirical disparity values are, generally speaking, in agreement with our benchmark models; however, the percentage of nodes for which $Q(Y_i \geq Y_i^*) < 0.05$, for our usual four snapshots, amounts at $0\%, 3.0\%, 9.1\%, 11\%$ for the UBCM-induced CReM$_\text{A}$ model and at $0\%, 5.6\%, 15\%, 17\%$ for the deterministic CReM$_\text{A}$ model: in other words, the percentage of nodes whose empirical disparity is significantly larger than predicted by one of the two null models considered here is rising throughout the entire BLN history - i.e. its vertices increasingly 'favor' some of the links surrounding them.

the $y = -x$ line, an evidence indicating that some kind of 'excess concentration' of weight (in one or more links) is indeed present - a tendency which is particularly evident for nodes with smaller degree.

Let us now compare the empirical disparity values with the predictions of the null models defined within our CReM$_\text{A}$ framework. To this aim, let us explicitly calculate the expected value of disparity, that reads

$$\begin{aligned}\langle Y_i \rangle &\simeq \frac{\sum_{j(\neq i)=1}^{N} \langle w_{ij}^2 \rangle}{\langle s_i^2 \rangle} \\ &= \frac{\sum_{j(\neq i)=1}^{N} \left(\text{Var}[w_{ij}] + \langle w_{ij} \rangle^2\right)}{\text{Var}[s_i] + \langle s_i \rangle^2} = \frac{\text{Var}[s_i] + \sum_{j(\neq i)=1}^{N} \langle w_{ij} \rangle^2}{\text{Var}[s_i] + s_i^2}, \quad \forall\, i\end{aligned} \tag{4.24}$$

where

$$\langle w_{ij} \rangle = \frac{p_{ij}}{\beta_i + \beta_j}, \quad \forall\, i < j \tag{4.25}$$

and

$$\text{Var}[s_i] = \sum_{\substack{j=1 \\ (j\neq i)=1}}^{N} \text{Var}[w_{ij}] = \sum_{\substack{j=1 \\ (j\neq i)=1}}^{N} \frac{p_{ij}}{(\beta_i + \beta_j)^2}, \quad \forall\, i \tag{4.26}$$

(naturally, for the the present analysis we have considered both the case $p_{ij} \equiv p_{ij}^{\text{UBCM}}, \forall\, i < j$ and the case $p_{ij} \equiv a_{ij}, \forall\, i < j$). As fig. 28 shows, disparity is, generally speaking, in agreement with our benchmark models. However, the calculation of the percentage of nodes for which $Q(Y_i \geq Y_i^*) < 0.05$, for our usual four snapshots, reveals it to be 0%, 3.0%, 9.1%, 11% for the UBCM-induced CReM$_\text{A}$ model and $0\%, 5.6\%, 15\%, 17\%$ for the deterministic CReM$_\text{A}$ model: in other words, the percentage of nodes whose empirical disparity is significantly larger than predicted by one of our two null models is rising throughout the entire BLN history. This evidence suggests that, as the BLN evolves, its vertices treat their neighbors less and less equally: indeed, they seem to place weights in a way that increasingly 'favors' some of the links surrounding them - a result that

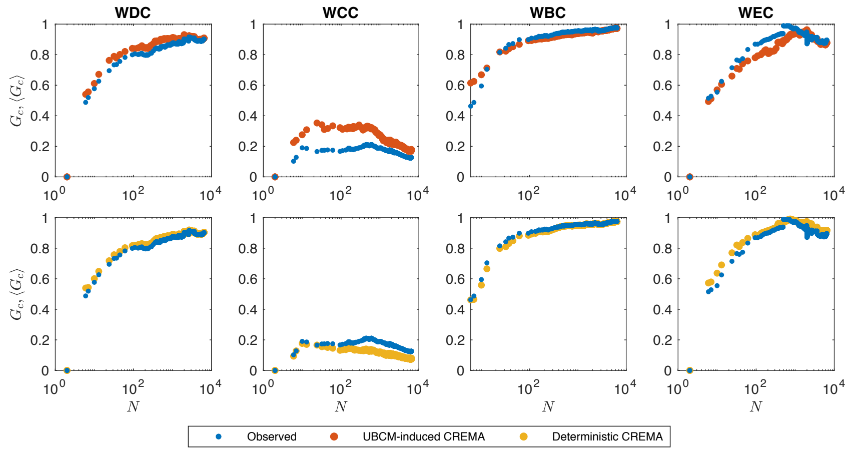


Figure 29: Evolution of the Gini coefficient of the empirical, weighted, centrality values (blue dots) and their expected counterparts, under the UBCM-induced CReM$_A$ model (red dots, upper panels) and the deterministic CReM$_A$ model (yellow dots, bottom panels). The rise of the Gini coefficient for the weighted degree, betweenness and eigenvector centrality points out that the distribution of centrality measures becomes increasingly uneven while the flat trend characterizing the evolution of the closeness centrality confirms what has been observed in the purely binary case: the aforementioned 'hubs' ensure the vast majority of nodes to be reachable (hence, to be close to each other) quite easily. Generally speaking, our null models tend to significantly underestimate both the weighted betweenness centrality and the weighted eigenvector centrality, signalling the presence of fewer-nodes-with-heavier-connections than predicted by chance.

remains true even when a null model constraining the entire topology of the BLN is employed[4].

[4]To be noticed that our null models also underestimate disparity values: however, the corresponding percentages (amounting at 3.2%, 4.3%, 5.8%, 5.2% for the UBCM-induced CReM$_A$ model and at 12.8%, 10.8%, 13.0%, 13.6% for the deterministic CReM$_A$ model, for our usual four snapshots), are typically lower and not increasing.

4.4.4 Centrality

Let us now comment the results concerning the weighted centrality measures considered in the present work. As a general observation, the weighted cases are characterized by trends which are overall similar to the trends characterizing the binary cases. As already observed for the purely binary BLN structure, the evolution of the Gini index for most centrality measures points out the latter ones to grow (strongly) unevenly distributed throughout the entire BLN history. While the rise of the Gini coefficient for the weighted degree, betweenness and eigenvector centrality measures suggests the appearance of nodes with 'heavy' connections - further confirmed by the strength distribution, which is a fat-tailed one - likely crossed by many paths and well connected between themselves, the flat trend characterizing the evolution of the closeness centrality confirms what has been already observed in the purely binary case, i.e. that the aforementioned 'hubs' ensure the vast majority of nodes to be reachable (hence, to be close to each other) quite easily.

Let us now compare the empirical trends of our four centrality measures with the ones predicted by our two null models. Figure 29 reveals that the UBCM-induced CReM$_A$ model tends to overestimate the values of the Gini index for the weighted degree and closeness centrality, i.e. the empirical weighted degree and closeness centrality measures are always significantly lower than their predicted counterparts. For what concerns the weighted betweenness centrality, instead, the percentage of snapshots for which $Q(G_{\text{WBC}} \geq G^*_{\text{WBC}}) < 0.05$ amounts at $\simeq 87\%$, i.e. the UBCM-induced CReM$_A$ model significantly underestimates the weighted betweenness centrality for $\simeq 87\%$ of the total number of snapshots. Analogously, the same null model tends to underestimate the values of the Gini index for the weighted eigenvector centrality roughly one third of the times: in fact, the percentage of snapshots for which $Q(G_{\text{WEC}} \geq G^*_{\text{WEC}}) < 0.05$ amounts at $\simeq 33\%$. Interestingly, the empirical WBC and WEC values are compatible with the predictions output by the UBCM-induced CReM$_A$ model, on the 'remaining' snapshots.

The deterministic CReM$_A$ model, instead, performs slightly better in

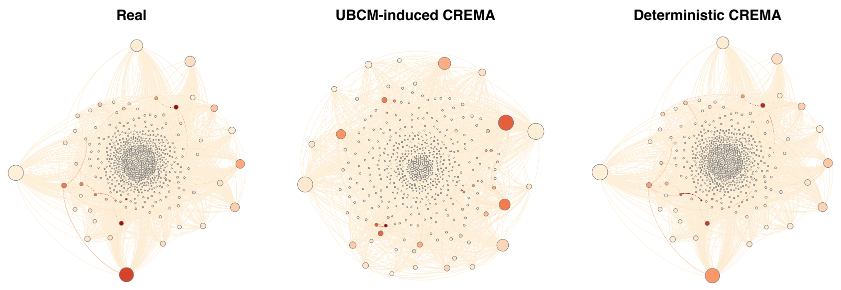


Figure 30: Comparison between the BLN on day 20-03-2018, a configuration generated by the UBCM-induced CReM$_A$ model and a configuration generated by the deterministic CReM$_A$ model for the same day. The latter one distributes weights more evenly than observed, hence underestimating disparity and letting the strength distribution become wider: this has an interesting consequence, i.e. letting the size of the core become larger than observed, under this model - it amounts at the 12%, the 27% and the 21% of the total number of nodes on the considered day - while still allowing the unevenness of the WEC distribution rise.

reproducing the centrality patterns characterizing the BLN: in fact, while it still overestimates the Gini index for the weighted degree centrality, the percentage of snapshots for which $Q(G_{\text{WCC}} \geq G^*_{\text{WCC}}) < 0.05$ amounts at $\simeq 96\%$, i.e. the deterministic CReM$_A$ model significantly underestimates the weighted closeness centrality for $\simeq 96\%$ of the total number of snapshots. For what concerns the weighted betweenness centrality, the percentage of snapshots for which $Q(G_{\text{WBC}} \geq G^*_{\text{WBC}}) < 0.05$ amounts at $\simeq 50\%$, i.e. the deterministic CReM$_A$ model significantly underestimates the weighted betweenness centrality roughly half of the times. Lastly, for what concerns the weighted eigenvector centrality, the percentage of snapshots for which $Q(G_{\text{WEC}} \leq G^*_{\text{WEC}}) < 0.05$ amounts at $\simeq 83\%$, i.e. the deterministic CReM$_A$ model significantly overestimates the weighted eigenvector centrality for $\simeq 83\%$ of the total number of snapshots. The deterministic CReM$_A$ model distributes weights more evenly than observed, hence underestimating disparity and letting the strength distribution become wider: this has an interesting consequence, i.e. letting the

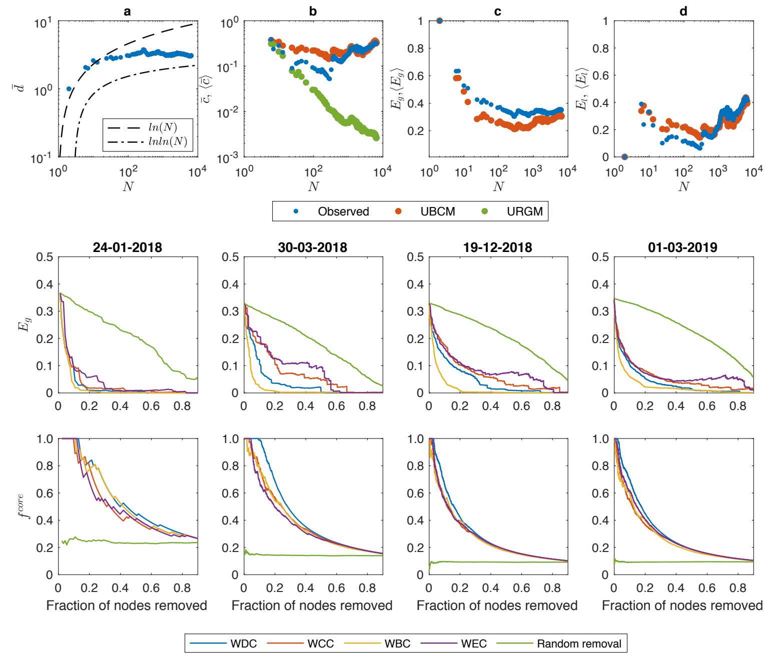


Figure 31: Upper panels: (a) evolution of the BLN average path length $\bar{d}$ and of the functions $\ln N$ and $\ln\ln N$; (b) evolution of the empirical average clustering coefficient $\overline{\text{BCC}}$ (blue dots) and of its expected value $\langle\overline{\text{BCC}}\rangle$ under the URGM (green dots) and the UBCM (red dots); (c) evolution of the empirical global efficiency E_g (blue dots) and of its expected value $\langle E_g\rangle$ under the UBCM (red dots); (d) evolution $\langle E_l\rangle$ under the UBCM (red dots). The BLN is indeed characterized by a small-world structure; moreover, while it has been always more-globally-efficient-than-expected under the UBCM, it has 'recently' become also more-locally-efficient-than-expected under the same null model. Middle panels: evolution of the BLN global efficiency, for our usual four snapshots, when nodes are removed either randomly (green trend) or sequentially, after having been sorted in decreasing order of weighted degree (blue trend), closeness (red trend), betweenness (yellow trend) and eigenvector (purple trend) centrality. The trends above characterize a robust-yet-fragile architecture: robust against the random removal of nodes but fragile against the targeted removal of nodes (e.g. an attack). Bottom panels: percentage of core nodes found within the set of nodes removed according to one of the two aforementioned criteria.

size of the core under this model become larger than observed - likely, because nodes with relatively low strength become, now, part of the core - while still allowing the unevenness of the WEC distribution rise.

Overall, these results point out a behavior that is not reproducible by just enforcing the degree and the strength sequences, irrespectively from the chosen index: in particular, the behavior of the weighted betweenness centrality points out that both null models - even if to a different extent - predict a more-even-than-observed structure.

4.4.5 Small-world -ness

The evidence that the BLN structure is more-centralized-than-expected rises an interesting question, i.e. if the BLN is small-world or not. From a purely empirical perspective, answering this question amounts at checking the behavior of the average path length, $\overline{d}$, and that of the average clustering coefficient, $\overline{\mathrm{BCC}} = \sum_i \mathrm{BCC}_i/N$ [136, 137, 138].

Figure 31 shows the results of these two analyses: while the evolution of $\overline{d}$ is described quite accurately by the function $\ln N$ during the first snapshots of the BLN history, its trend has progressively become more and more similar to the smoothest one characterizing the function $\ln \ln N$ - which has reached the value $\simeq 3.5$ on the snapshot with 10^4 nodes. For what concerns the average clustering coefficient, one needs to compare it with the value predicted by the URGM, i.e. the null model prescribing to link each pair of nodes with the same probability $p = 2L/N(N-1)$: as fig. 31 shows, the URGM significantly underestimates the average clustering coefficient throughout the entire BLN history; taken together, there results indicate that the BLN is indeed small-world. On the other hand, the UBCM overestimates $\overline{\mathrm{BCC}} = \sum_i \mathrm{BCC}_i/N$ during the first half of its history (for $\simeq 40\%$ of the total number of snapshots), thus signaling a tendency of our system to avoid closing paths among triples of nodes.

An alternative way of testing small-world -ness is that of checking the behavior of efficiency. Overall, the global efficiency amounts at $E_g \simeq 0.4$ and it is significantly underestimated by the UBCM throughout the entire history of the BLN. This indicates that the BLN exchanges information more-efficiently-than-predicted by a null model retaining only the information provided by degrees and can be a consequence of the presence of hubs crossed by many paths that shorten the topological distance between (any pair of) nodes. These results suggest that the BLN has progressively self-organized to 'keep the overall distances low'.

What about efficiency from a local point of view? For what concerns the local efficiency, the percentage of nodes for which $Q(E_l \geq E_l^*) < 0.05$ amounts at 75%: hence, the UBCM significantly underestimates it for a large portion of the BLN snapshots - as evident from fig. 31, the most recent ones. As the local efficiency $E(\mathbf{G}_i)$ provides information about how efficient the communication between the first neighbors of node i is, upon its removal, our results seem to indicate that the BLN is becoming more and more 'fault tolerant' than its randomized counterpart (interestingly, it appeared to be much more fragile during the first half of its history). This result can be understood by imagining that a larger number of redundant connections has been established, among nodes, in the more recent snapshots of the BLN history - whence the rise of the average clustering coefficient as well.

As an additional exercise, let us inspect the evolution of the BLN global efficiency as nodes are removed either randomly or sequentially, after they have been sorted in decreasing order of weighted degree, closeness, betweenness and eigenvector centrality. The results of our exercise are shown in fig. 31. The depicted trends are compatible with a *robust-yet-fragile* architecture, i.e. a topological structure that is robust against a random removal of nodes but fragile against a targeted removal of nodes (e.g. an attack) - or, more correctly, more robust against a random node removal than against a targeted node removal: notice how steeper the decrease of E_g is in the second case; moreover, removing nodes according to their WBC reduces the BLN global efficiency to the largest extent (for the vast majority of snapshots, larger than removing nodes according to their WDC, WCC and WEC). The same figure also shows that the nodes whose removal brings the most severe damages to the BLN are those belonging to the core (see the next paragraph), whose size shrinks

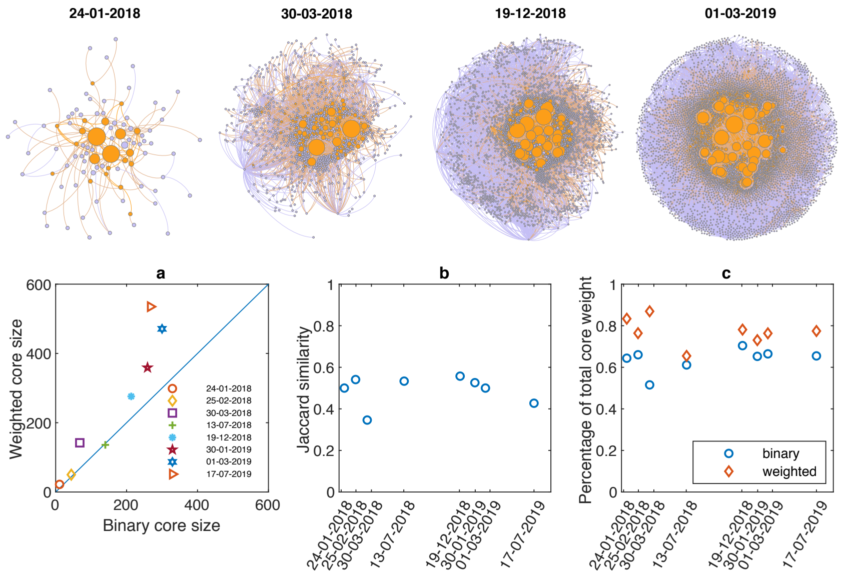


Figure 32: Upper panels: core-periphery structure, as revealed by the weighted surprise, $\mathscr{W}_{/\!/}$, for our four usual snapshots (core nodes are colored in orange and periphery nodes are colored in purple); the size of the nodes is proportional to their strength: hence, the nodes constituting the core of the network are precisely those with a larger strength. Bottom panels: (a) values of the size of the binary core scattered versus the values of the size of the weighted core; (b) evolution of (the percentage of) the overlap between the set of nodes belonging to the binary core and the set of nodes belonging to the weighted one, estimated via the Jaccard similarity, on a selected subset of snapshots of the BLN; evolution of the percentage of the total network weight embodied by 'core connections', amounting at $\simeq 68\%$ in the binary case (blue circles) and at $\simeq 77\%$ in the weighted one (red diamonds).

from $\gtrsim 20\%$ to $\simeq 10\%$ of the total number of nodes.

4.4.6 Analysis of the BLN mesoscale structure

The result concerning the underestimation, by our null models, of the Gini index for the weighted eigenvector centrality - i.e. the presence of well connected nodes which, in turn, are also well connected among them - lets us suppose the BLN to be characterized by a statistically-significant core-periphery structure: here, however, we are interested to reveal the presence of a weighted core-periphery structure, i.e. a kind of mesoscale organization where core nodes are the ones sharing the 'heaviest' connections - and not just those with 'many' connections.

To this aim, we adopt the approach described in the previous chapter and based upon the surprise formalism. In particular, the evolution of the weighted bimodular surprise, $\mathscr{W}_{/\!/}$, across the entire BLN history reveals that the statistical significance of the recovered core-periphery structure increases, a result leading to the conclusion that the description of the BLN structure provided by such a model becomes more and more accurate as the network evolves. Figure 32 shows the detected core-periphery structure on the snapshots depicted in the same figure: the nodes identified as belonging to the core are colored in orange while those identified as belonging to the periphery are colored in purple. Notice also that we have drawn the node size proportionally to the node strength: hence, larger nodes, i.e. the ones sharing the 'heaviest' connections, are precisely those constituting the core of the network.

Let us now check the correspondence between the nodes in the core (whose size will be indicated as N_{core}) and those with a large weighted eigenvector centrality, by ranking the nodes in decreasing order of WEC and checking the percentage of top N_{core} nodes that belong to the core as well: it amounts at $56\%, 60\%, 57\%, 62\%$, for our usual four snapshots. Let us also compare the composition of the purely binary core - detected in [55] - with that of the weighted core: as fig. 32 shows, a nice correspondence between the size of binary core and that of the weighted one indeed exists although, from a certain moment of the BLN history on, the binary core seems to 'grow slower' than the weighted one which, instead, enlarges to reach a size of $\simeq 600$ nodes: this further confirms that the nodes with a 'large' strength, revealed by surprise as the most central ones, do not necessarily coincide with those having a 'large' degree.

The evolution of (the percentage of) the overlap between the set of nodes belonging to the binary core and the set of nodes belonging to the weighted one further confirms that the two sets do not coincide perfectly, although the Jaccard similarity steadily points out a $\simeq 60\%$ of overlap: in other words, 60% of the nodes belong to both cores - likely, those hubs whose degree *and* strength are large enough to justify their coreness in both senses; similarly, the percentage of the total network weight embodied by 'core connections' amounts at $\simeq 68\%$ in the binary case and at $\simeq 77\%$ in the weighted one.

4.5 Discussion

The analysis of the binary BLN structure carried out in [55] has revealed a system whose topology has become increasingly characterized by star-like structures, whose centers are constituted by 'hubs' to which many nodes having a (much) small(er) degree, in turn, attach. Such a structure - whose disassortativity is confirmed by scattering the ANND values versus the degrees - could explain the more-than-expected level of unevenness characterizing the betweenness and the eigenvector centralization indices, suggesting them to be due to the emergence of channel-switching nodes - apparently, an unavoidable consequence of the way BLN is designed: on the one hand, as longer routes are more expensive, any two BLN users will search for a short(est) path; on the other, nodes have the incentive to become as central as possible, in order to maximize the transaction fees they may earn.

The tendency to centralization is observable even when considering weighted quantities, as the percentage of nodes whose connections embody the 51% of the total weight progressively reduces and the Gini coefficient of several (weighted) centrality measures steadily increases throughout the entire BLN history. This clearly points out the co-existence of nodes playing deeply different 'structural' roles, with 'many' peripheral vertices co-existing with 'few' core ones; if, on the one hand, this structure allows the global efficiency to achieve a large value (i.e. hubs facilitate the global exchange of information, being at the origin of another structural BLN peculiarity, i.e. its small-world -ness), on the other it highlights the tendency of the BLN architecture to become increasingly 'less distributed', a process having the undesirable consequence of making it increasingly fragile towards failures and attacks.

Distinguishing between the two is crucial, in order to properly understand the BLN robustness to 'damages'. While resilience towards failures can be tested by looking at how the global efficiency 'reacts' to random node removal, resilience towards attacks can, instead, be quantified by implementing targeted removals of the 'most important' nodes. To this aim, we have ranked nodes in decreasing order of weighted degree, closeness, betweenness and eigenvector centrality and removed them, sequentially: the global efficiency drops rapidly after few (core) nodes are deleted - in fact, for almost all snapshots, removing just *one top node* (according to any of the aforementioned criteria) is enough to disconnect the graph. Moreover, since top nodes are likely to be part of the core - whose size shrinks from $\gtrsim 20\%$ to $\simeq 10\%$ of the total number of nodes - our results indicate that the vertices belonging to it are precisely those whose removal causes the major structural damages. Random failures, instead, cause the decrease of E_g to be much less steep: taken together, the results above seem to indicate that the BLN topology is an example of *robust-yet-fragile* architecture, i.e. a structure that is robust against a random node removal but fragile against a targeted node removal (e.g. an attack).

Chapter 5

Conclusions

The definition and correct implementation of null models is a crucial issue in network analysis.

In the first chapter of the present thesis we have implemented and compared three algorithms for the numerical optimization of the likelihood function of a bunch of ERGs[1], with the aim of finding the one performing best (i.e. being both accurate and fast) for each model. Such an issue is particularly relevant upon considering that the computational cost of such a task - which involves the resolution of a system of $O(N)$ non-linear, coupled equations where N represents the number of nodes of the network under analysis - increases as the network size becomes larger. What emerges from our results is that there is no unique method which is both accurate and fast for all models on all networks: under this respect, performance is just a trade-off between accuracy and speed. Overall, while Newton's method seems to perform best on either relatively small or greatly reducible networks, the fixed-point method must be preferred for large, non-reducible configurations. The performance of the quasi-Newton method often lies in-between the performances of the two methods above, by achieving an accuracy that is larger than the one achieved by the fixed-point algorithm while requiring less time that the

[1]Specifically, the ones defined by local constraints: this choice has been driven by the evidence that they are the most commonly employed ones for tasks as different as network reconstruction, pattern detection, graph enumeration.

one needed by Newton's method[2].

In this respect, future work concerns the application of the three, aforementioned, numerical recipes to solve the models that have not been considered here. For what concerns the set of purely binary constraints, the ones defining the *Reciprocal Binary Configuration Model* (RBCM) deserve to be mentioned; for what concerns the 'mixed' constraints, instead, the CReM framework is versatile enough to accommodate a quite large number of variants. In the present work, we have 'limited' ourselves to combine the UBCM and the DBCM with (conditional) distributions of continuous weights: a first, obvious, generalization is that of considering the discrete versions of such models; a second one concerns the continuous versions of the UECM and of the DECM. Naturally, both generalizations call for the extension of the 'NEMTROPY' package as well.

In the second chapter of the present thesis we have explored the 'power' of the hypergeometric formalism to carry out the detection of mesoscale network structures: remarkably, it allows proper statistical tests to be definable in order to reveal the presence of modular (i.e. communities) and 'bimodular' (i.e. core-periphery and bipartite) structures on any kind of network, be it binary or weighted, undirected or directed. As we have stressed there, we believe surprise to belong to the class of algorithms implementing proper tests of hypothesis between two competing alternatives (e.g. the RGM and the SBM, the WRG and the WSBM).

Although our surprise-based approach is powerful and versatile, its downside is represented by the specific kinds of tests that are induced by the optimization of surprise-like score functions: comparisons between benchmarks 'ignoring' the local structure of nodes (i.e. their degree) are, in fact, carried out. While this seems to be perfectly reasonable when considering core-periphery structures, this is no longer true for the community detection task: indeed, as it has been noticed elsewhere, ignoring degrees may be at the origin of the large number of singletons output by surprise as assigning nodes with few neighbors to larger clusters may

[2]Deviations from this (over simplified) picture are, however, clearly visible.

be disfavored from a statistical point of view. The observations above call for the 'extension' of the hypergeometric formalism we have studied here to include more refined benchmarks as the ones constraining the entire degree sequence.

Overall, our observations suggest a strategy to handle mesoscale structures detection on real-world networks: as the information on the presence of a possible, 'true' partition is rarely available, a good strategy may be that of running different algorithms on the same empirical networks, check the consistency of their output and combine them, e.g. by taking the overlap - in a way that is reminiscent of multi-model inference.

Lastly, in the fourth chapter we have explored the application of the methods described in chapters 2 and 3 to the Bitcoin Lightning Network - which represents an attempt to overcome one of the main limitations of the Bitcoin technological design, i.e. scalability: at the moment, only a limited amount of transactions per second can be processed by Bitcoin, a major shortcoming preventing the adoption of this payment system at a global scale.

Most of the contributions about the Bitcoin Lightning Network, so far, have left its weighted structure largely unexplored. With our work, we have tried to fill this gap, by studying the weighted properties of the BLN daily snapshot representation, at the micro-, meso- and macro-scale. Benchmarking quantities such as the centrality indices highlights the latter ones to be significantly different from what it is expected expected under any of the considered null models: in particular, the underestimation of the Gini index by the CReM_A model suggested the BLN to be characterized by a (statistically-significant) core-periphery structure and the optimization of the weighted, bimodular surprise has later confirmed this intuition.

The tendency to centralization of the BLN has been revealed by comparing the empirical structure of the latter with properly-defined benchmark models, a result further confirming the importance of the theoretical work behind data analysis, devoted to (analytically) define and (numerically) solve *models*.

Appendix A

The Hessian matrix for ERGs with local constraints

A.1 Computing the Hessian matrix

As we showed in the main text, the Hessian matrix of our likelihood function is 'minus' the covariance matrix of the constraints, i.e.

$$H_{ij} = \frac{\partial^2 \mathscr{L}(\vec{\theta})}{\partial\theta_i \partial\theta_j} = -\mathrm{Cov}[C_i, C_j], \quad i, j = 1 \dots M \tag{A.1}$$

interestingly, a variety of alternative methods exists to explicitly calculate the generic entry H_{ij}, i.e. 1) taking the second derivatives of the likelihood function characterizing the method under analysis, 2) taking the first derivatives of the expectation values of the constraints characterizing the method under analysis, 3) calculating the moments of the pair-specific probability distributions characterizing each method.

A.1.1 UBCM

The Hessian matrix for the UBCM is an $N \times N$ symmetric table with entries reading

$$H_{\text{UBCM}} = \begin{cases} \frac{\partial^2 \mathscr{L}_{\text{UBCM}}}{\partial \theta_i^2} = \text{Var}[k_i] = \sum_{\substack{j=1 \\ (j \neq i)}}^N p_{ij}(1-p_{ij}), \quad \forall\, i \\ \frac{\partial^2 \mathscr{L}_{\text{UBCM}}}{\partial \theta_i \theta_j} = \text{Cov}[k_i, k_j] = p_{ij}(1-p_{ij}), \quad \forall\, i \neq j \end{cases} \tag{A.2}$$

where $p_{ij} \equiv p_{ij}^{\text{UBCM}}$. Notice that $\text{Var}[k_i] = \sum_{j(\neq i)=1}^N \text{Cov}[k_i, k_j], \forall\, i$.

A.1.2 UECM

The Hessian matrix for the UECM is a $2N \times 2N$ symmetric table that can be further subdivided into four blocks (each of which with dimensions $N \times N$). In order to save space, the expressions indexed by the single subscript i will be assumed as being valid $\forall\, i$, while the ones indexed by a double subscript i, j will be assumed as being valid $\forall\, i \neq j$. The entries of the diagonal blocks read

$$H_{\text{UECM}} = \begin{cases} \frac{\partial^2 \mathscr{L}_{\text{UECM}}}{\partial \alpha_i^2} = \text{Var}[k_i] = \sum_{\substack{j=1 \\ (j \neq i)}}^N p_{ij}(1-p_{ij}) \\ \frac{\partial^2 \mathscr{L}_{\text{UECM}}}{\partial \alpha_i \alpha_j} = \text{Cov}[k_i, k_j] = p_{ij}(1-p_{ij}) \end{cases} \tag{A.3}$$

and

$$H_{\text{UECM}} = \begin{cases} \frac{\partial^2 \mathscr{L}_{\text{UECM}}}{\partial \beta_i^2} = \text{Var}[s_i] = \sum_{\substack{j=1 \\ (j \neq i)}}^N \frac{p_{ij}(1-p_{ij}+e^{-\beta_i-\beta_j})}{(1-e^{-\beta_i-\beta_j})^2} \\ \frac{\partial^2 \mathscr{L}_{\text{UECM}}}{\partial \beta_i \beta_j} = \text{Cov}[s_i, s_j] = \frac{p_{ij}(1-p_{ij}+e^{-\beta_i-\beta_j})}{(1-e^{-\beta_i-\beta_j})^2} \end{cases} \tag{A.4}$$

where $p_{ij} \equiv p_{ij}^{\text{UECM}}$. On the other hand, the entries of the off-diagonal blocks read

$$H_{\text{UECM}} = \begin{cases} \frac{\partial^2 \mathscr{L}_{\text{UECM}}}{\partial \alpha_i \partial \beta_i} = \text{Cov}[k_i, s_i] = \sum_{\substack{j=1 \\ (j \neq i)}}^N \frac{p_{ij}(1-p_{ij})}{1-e^{-\beta_i-\beta_j}} \\ \frac{\partial^2 \mathscr{L}_{\text{UECM}}}{\partial \alpha_i \partial \beta_j} = \text{Cov}[k_i, s_j] = \frac{p_{ij}(1-p_{ij})}{1-e^{-\beta_i-\beta_j}} \end{cases} \tag{A.5}$$

with $p_{ij} \equiv p_{ij}^{\text{UECM}}$.

A.1.3 DECM

The Hessian matrix for the DECM is a $4N \times 4N$ symmetric table that can be further subdivided into four blocks (each of which with dimensions $N \times N$). As for the UECM, in order to save space, the expressions indexed by the single subscript i will be assumed as being valid $\forall i$, while the ones indexed by a double subscript i, j will be assumed as being valid $\forall\, i \neq j$. The entries of the diagonal blocks read

$$H_{\text{DECM}} = \begin{cases} \frac{\partial^2 \mathscr{L}_{\text{DECM}}}{\partial \alpha_i^2} = \text{Var}[k_i^{out}] = \sum_{\substack{j=1 \\ (j \neq i)}}^N p_{ij}(1 - p_{ij}) \\ \frac{\partial^2 \mathscr{L}_{\text{DECM}}}{\partial \alpha_i \alpha_j} = \text{Cov}[k_i^{out}, k_j^{out}] = 0 \end{cases} \tag{A.6}$$

and

$$H_{\text{DECM}} = \begin{cases} \frac{\partial^2 \mathscr{L}_{\text{DECM}}}{\partial \beta_i^2} = \text{Var}[k_i^{in}] = \sum_{\substack{j=1 \\ (j \neq i)}}^N p_{ji}(1 - p_{ji}) \\ \frac{\partial^2 \mathscr{L}_{\text{DECM}}}{\partial \beta_i \beta_j} = \text{Cov}[k_i^{in}, k_j^{in}] = 0 \end{cases} \tag{A.7}$$

and

$$H_{\text{DECM}} = \begin{cases} \frac{\partial^2 \mathscr{L}_{\text{DECM}}}{\partial \gamma_i^2} = \text{Var}[s_i^{out}] = \sum_{\substack{j=1 \\ (j \neq i)}}^N \frac{p_{ij}(1-p_{ij}+e^{-\gamma_i-\delta_j})}{(1-e^{-\gamma_i-\delta_j})^2} \\ \frac{\partial^2 \mathscr{L}_{\text{DECM}}}{\partial \gamma_i \gamma_j} = \text{Cov}[s_i^{out}, s_j^{out}] = 0 \end{cases} \tag{A.8}$$

and

$$H_{\text{DECM}} = \begin{cases} \frac{\partial^2 \mathscr{L}_{\text{DECM}}}{\partial \delta_i^2} = \text{Var}[s_i^{in}] = \sum_{\substack{j=1 \\ (j \neq i)}}^N \frac{p_{ji}(1-p_{ji}+e^{-\gamma_j-\delta_i})}{(1-e^{-\gamma_j-\delta_i})^2} \\ \frac{\partial^2 \mathscr{L}_{\text{DECM}}}{\partial \delta_i \delta_j} = \text{Cov}[s_i^{in}, s_j^{in}] = 0 \end{cases} \tag{A.9}$$

where $p_{ij} \equiv p_{ij}^{\text{DECM}}$. On the other hand, the entries of the off-diagonal blocks read

$$H_{\text{DECM}} = \begin{cases} \frac{\partial^2 \mathscr{L}_{\text{DECM}}}{\partial \alpha_i \partial \beta_i} = \text{Cov}[k_i^{out}, k_i^{in}] = 0 \\ \frac{\partial^2 \mathscr{L}_{\text{DECM}}}{\partial \alpha_i \partial \beta_j} = \text{Cov}[k_i^{out}, k_j^{in}] = p_{ij}(1 - p_{ij}) \end{cases} \tag{A.10}$$

and

$$H_{\text{DECM}} = \begin{cases} \frac{\partial^2 \mathscr{L}_{\text{DECM}}}{\partial \alpha_i \partial \gamma_i} = \text{Cov}[k_i^{out}, s_i^{out}] = \sum_{\substack{j=1 \\ (j \neq i)}}^{N} \frac{p_{ij}(1-p_{ij})}{1-e^{-\gamma_i - \delta_j}} \\ \frac{\partial^2 \mathscr{L}_{\text{DECM}}}{\partial \alpha_i \partial \gamma_j} = \text{Cov}[k_i^{out}, s_j^{out}] = 0 \end{cases} \tag{A.11}$$

and

$$H_{\text{DECM}} = \begin{cases} \frac{\partial^2 \mathscr{L}_{\text{DECM}}}{\partial \alpha_i \partial \delta_i} = \text{Cov}[k_i^{out}, s_i^{in}] = 0 \\ \frac{\partial^2 \mathscr{L}_{\text{DECM}}}{\partial \alpha_i \partial \delta_j} = \text{Cov}[k_i^{out}, s_j^{in}] = \frac{p_{ij}(1-p_{ij})}{1-e^{-\gamma_i - \delta_j}} \end{cases} \tag{A.12}$$

and

$$H_{\text{DECM}} = \begin{cases} \frac{\partial^2 \mathscr{L}_{\text{DECM}}}{\partial \beta_i \partial \gamma_i} = \text{Cov}[k_i^{in}, s_i^{out}] = 0 \\ \frac{\partial^2 \mathscr{L}_{\text{DECM}}}{\partial \beta_i \partial \gamma_j} = \text{Cov}[k_i^{in}, s_j^{out}] = \frac{p_{ji}(1-p_{ji})}{1-e^{-\gamma_j - \delta_i}} \end{cases} \tag{A.13}$$

and

$$H_{\text{DECM}} = \begin{cases} \frac{\partial^2 \mathscr{L}_{\text{DECM}}}{\partial \beta_i \partial \delta_i} = \text{Cov}[k_i^{in}, s_i^{in}] = \sum_{\substack{j=1 \\ (j \neq i)}}^{N} \frac{p_{ji}(1-p_{ji})}{1-e^{-\gamma_j - \delta_i}} \\ \frac{\partial^2 \mathscr{L}_{\text{DECM}}}{\partial \beta_i \partial \delta_j} = \text{Cov}[k_i^{in}, s_j^{in}] = 0 \end{cases} \tag{A.14}$$

and

$$H_{\text{DECM}} = \begin{cases} \frac{\partial^2 \mathscr{L}_{\text{DECM}}}{\partial \gamma_i \partial \delta_i} = \text{Cov}[s_i^{out}, s_i^{in}] = 0 \\ \frac{\partial^2 \mathscr{L}_{\text{DECM}}}{\partial \gamma_i \partial \delta_j} = \text{Cov}[s_i^{out}, s_j^{in}] = \frac{p_{ij}(1-p_{ij}+e^{-\gamma_i - \delta_j})}{(1-e^{-\gamma_i - \delta_j})^2} \end{cases} \tag{A.15}$$

with $p_{ij} \equiv p_{ij}^{\text{DECM}}$.

A.1.4 Two-step models

The Hessian matrix for the undirected two-step model considered here is an $N \times N$ symmetric table reading

$$H^{\text{und}}_{\text{CReM}}=\begin{cases}\text{Var}[s_i]=\sum_{\substack{j=1\\(j\neq i)}}^{N}\frac{f_{ij}}{(\theta_i+\theta_j)^2}, & \forall\, i\\ \text{Cov}[s_i,s_j]=\frac{f_{ij}}{(\theta_i+\theta_j)^2}, & \forall\, i\neq j\end{cases}\tag{A.16}$$

where f_{ij} is given. In the directed case, instead, the Hessian matrix for the two-step model considered here is a $2N\times 2N$ symmetric table that can be further subdivided into four $N\times N$ blocks whose entries read

$$H^{\text{dir}}_{\text{CReM}}\begin{cases}\text{Var}[s_i^{out}]=\sum_{\substack{j=1\\(j\neq i)}}^{N}\frac{f_{ij}}{(\alpha_i+\beta_j)^2}, & \forall\, i\\ \text{Var}[s_i^{in}]=\sum_{\substack{j=1\\(j\neq i)}}^{N}\frac{f_{ji}}{(\alpha_j+\beta_i)^2}, & \forall\, i\\ \text{Cov}[s_i^{out},s_j^{in}]=\frac{f_{ij}}{(\alpha_i+\beta_j)^2}, & \forall\, i\neq j\end{cases}\tag{A.17}$$

while $\text{Cov}[s_i^{out},s_i^{in}]=\text{Cov}[s_i^{out},s_j^{out}]=\text{Cov}[s_i^{in},s_j^{in}]=0$ and f_{ij} is given.

A.2 Fixed-point method in the multivariate case

Equation (2.25) can be written as

$$\theta_i^{(n)}=G_i(\vec{\theta}^{(n-1)}),\quad i=1\dots N\tag{A.18}$$

For the sake of illustration, let us discuss it for the UBCM case. In this particular case, the set of equations above can be rewritten as

$$\theta_i^{(n)}=-\ln\left[\frac{k_i(\mathbf{A}^*)}{\sum_{\substack{j=1\\(j\neq i)}}^{N}\left(\frac{e^{-\theta_j^{(n-1)}}}{1+e^{-\theta_i^{(n-1)}-\theta_j^{(n-1)}}}\right)}\right],\quad i=1\dots N\tag{A.19}$$

Since all components of the map $\mathbf{G}$ are continuous on $\mathbb{R}^N$, the map itself is continuous on $\mathbb{R}^N$. Hence, a fixed point exists. Let us now consider its Jacobian matrix and check the magnitude of its elements. In the UBCM case, one finds that

$$\begin{aligned}\frac{\partial G_i}{\partial \theta_i} &= \frac{\sum_{\substack{j=1\\(j\neq i)}}^{N} \frac{e^{-\theta_i-2\theta_j}}{(1+e^{-\theta_i-\theta_j})^2}}{\sum_{\substack{j=1\\(j\neq i)}}^{N}\left(\frac{e^{-\theta_j}}{1+e^{-\theta_i-\theta_j}}\right)} = \frac{\sum_{\substack{j=1\\(j\neq i)}}^{N}\left(\frac{e^{-\theta_i-\theta_j}}{1+e^{-\theta_i-\theta_j}}\right)^2}{\sum_{\substack{j=1\\(j\neq i)}}^{N}\left(\frac{e^{-\theta_i-\theta_j}}{1+e^{-\theta_i-\theta_j}}\right)} \\ &= 1-\frac{\text{Var}[k_i]}{\langle k_i\rangle} = 1-\frac{\sum_{j\neq i}\text{Cov}[k_i,k_j]}{\langle k_i\rangle}, \quad \forall\, i\end{aligned} \tag{A.20}$$

and

$$\begin{aligned}\frac{\partial G_i}{\partial \theta_j} &= -\frac{\frac{e^{-\theta_j}}{(1+e^{-\theta_i-\theta_j})^2}}{\sum_{\substack{j=1\\(j\neq i)}}^{N}\left(\frac{e^{-\theta_j}}{1+e^{-\theta_i-\theta_j}}\right)} = -\frac{\frac{e^{-\theta_i-\theta_j}}{(1+e^{-\theta_i-\theta_j})^2}}{\sum_{\substack{j=1\\(j\neq i)}}^{N}\left(\frac{e^{-\theta_i-\theta_j}}{1+e^{-\theta_i-\theta_j}}\right)} \\ &= -\frac{\text{Cov}[k_i,k_j]}{\langle k_i\rangle}, \quad \forall\, i,j\end{aligned} \tag{A.21}$$

Let us notice that 1) each element of the Jacobian matrix is a continuous function $\mathbb{R}^N \to \mathbb{R}$ and that 2) the following relationships hold

$$\left|\frac{\partial G_i}{\partial \theta_j}\right| \leq 1, \quad \forall\, i,j \tag{A.22}$$

unfortunately, however, when multivariate functions are considered, the set of conditions above is not enough to ensure convergence to the fixed point for *any* choice of the initial value of the parameters. What is needed to be checked is the condition $||J_{\mathbf{G}}(\vec{\theta})|| < 1$, with J indicating the Jacobian of the map (i.e. the matrix of the first, partial derivatives above) and $||.||$ any natural matrix norm: the validity of such a condition has been numerically verified case by case.

Appendix B

Detecting mesoscale structures by surprise

B.1 Detecting binary mesoscale structures

As stressed in the main text, community detection on binary networks can be implemented by carrying out an exact statistical test whose function f must be identified with a binomial hypergeometric distribution. Hence, we can compactly write

$$f(l_\bullet) \equiv \mathrm{H}(l_\bullet|V, V_\bullet, L) = \frac{\prod_{i=\bullet,\circ}\binom{V_i}{l_i}}{\binom{V}{L}} = \frac{\binom{V_\bullet}{l_\bullet}\binom{V_\circ}{l_\circ}}{\binom{V}{L}} = \frac{\binom{V_\bullet}{l_\bullet}\binom{V-V_\bullet}{L-l_\bullet}}{\binom{V}{L}} \tag{B.1}$$

with a clear meaning of the symbols. The correctedness of such the definition of can be explicitly checked upon calculating its normalization via the Vandermonde's identity $\sum_{k=0}^{r}\binom{m}{k}\binom{n}{r-k} = \binom{m+n}{r}$. While the binomial coefficient $\binom{V_\bullet}{l_\bullet}$ enumerates the number of ways $l_\bullet$ links can be redistributed within communities, i.e. over the available $V_\bullet$ node pairs, the binomial coefficient $\binom{V-V_\bullet}{L-l_\bullet}$ enumerates the number of ways the remaining $L - l_\bullet$ links can be redistributed between communities, i.e. over the remaining $V - V_\bullet$ node pairs. The position above, in turn, induces the definition of the binary surprise, i.e. $\mathscr{S} \equiv \sum_{l_\bullet \geq l_\bullet^*} f(l_\bullet)$.

The generalization of the formalism above to detect 'bimodular' structure on binary networks is straightforward. It is enough to consider the multivariate (or multinomial) version of the hypergeometric distribution, i.e.

$$f(l_\bullet, l_\circ) \equiv \text{MH}(l_\bullet, l_\circ | V, V_\bullet, V_\circ, L)$$
$$= \frac{\prod_{i=\bullet,\circ,\top} \binom{V_i}{l_i}}{\binom{V}{L}} = \frac{\binom{V_\bullet}{l_\bullet}\binom{V_\circ}{l_\circ}\binom{V_\top}{l_\top}}{\binom{V}{L}} = \frac{\binom{V_\bullet}{l_\bullet}\binom{V_\circ}{l_\circ}\binom{V-(V_\bullet+V_\circ)}{L-(l_\bullet+l_\circ)}}{\binom{V}{L}}; \quad \text{(B.2)}$$

the correctedness of such a definition of can be checked upon calculating its normalization by applying the identity $\sum_{k=0}^{r}\binom{m}{k}\binom{n}{r-k} = \binom{m+n}{r}$ twice. The position above, in turn, induces the definition of the binary bimodular surprise, i.e. $\mathscr{S}_{/\!/} \equiv \sum_{l_\bullet \geq l_\bullet^*} \sum_{l_\circ \geq l_\circ^*} f(l_\bullet, l_\circ)$.

The asymptotic expression of $\mathscr{S}$ and $\mathscr{S}_{/\!/}$ can be derived by applying the Stirling approximation $n! \simeq \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$ to the binomial coefficients appearing into the definition of the versions of the surprise described above, leading to expressions like

$$\binom{V}{L} = \frac{V!}{L!(V-L)!} \simeq \frac{\sqrt{2\pi V}V^V e^{-V}}{\sqrt{2\pi L}L^L e^{-L}\sqrt{2\pi(V-L)}(V-L)^{V-L}e^{-(V-L)}}$$
$$= \frac{1}{\sqrt{2\pi V \frac{L}{V}\left(1-\frac{L}{V}\right)}} \cdot \frac{V^V}{L^L(V-L)^{V-L}}$$
$$= \frac{1}{\sqrt{2\pi V p(1-p)}} \cdot \frac{1}{\left(\frac{L}{V}\right)^L\left(1-\frac{L}{V}\right)^{V-L}}$$
$$= \frac{1}{\sqrt{2\pi V p(1-p)}} \cdot \frac{1}{p^L(1-p)^{V-L}} = \frac{1}{\sqrt{2\pi\sigma^2}\cdot \text{Ber}(V,L,p)} \quad \text{(B.3)}$$

where $\text{Ber}(x,y,z) = z^y(1-z)^{x-y}$ is a Bernoulli probability mass function, $z = \frac{y}{x}$ and $p = \frac{L}{V}$.

The time, in seconds, to find the corresponding optimal partitions is reported in tables 9 and 10: overall, denser networks require more time.

B.2 Detecting weighted mesoscale structures

When analyzing weighted networks, one needs to move from the binomial hypergeometric distribution to its negative counterpart, i.e. to

$$
\begin{aligned}
f(w_\bullet) &= \mathrm{NH}(w_\bullet|V+W,W,V_\bullet)\\
&= \frac{\prod_{i=\bullet,\circ}\binom{V_i+w_i-1}{w_i}}{\binom{V+W-1}{W}} = \frac{\binom{V_\bullet+w_\bullet-1}{w_\bullet}\binom{V_\circ+w_\circ-1}{w_\circ}}{\binom{V+W-1}{W}}\\
&= \frac{\binom{V_\bullet+w_\bullet-1}{w_\bullet}\binom{V-V_\bullet+(W-w_\bullet)-1}{W-w_\bullet}}{\binom{V+W-1}{W}}
\end{aligned}
\tag{B.4}
$$

with a clear meaning of the symbols. The soundness of the definition above can be checked upon calculating the normalization of the distribution $\mathrm{NH}(w_\bullet|V+W,W,V_\bullet)$ via the identity $\sum_{k=0}^{r}\binom{k+m}{k}\binom{n-m-k}{r-k}=\binom{n+1}{r}$: in fact

$$
\begin{aligned}
\sum_{w_\bullet=0}^{W}\frac{\binom{w_\bullet+V_\bullet-1}{w_\bullet}\binom{w_\circ+V_\circ-1}{w_\circ}}{\binom{V+W-1}{W}} &= \sum_{w_\bullet=0}^{W}\frac{\binom{w_\bullet+V_\bullet-1}{w_\bullet}\binom{W-w_\bullet+V-V_\bullet-1}{W-w\bullet}}{\binom{V+W-1}{W}}\\
&= \sum_{w_\bullet=0}^{W}\frac{\binom{w_\bullet+V_\bullet-1}{w_\bullet}\binom{(W+V-2)-(V_\bullet-1)-w_\bullet}{W-w_\bullet}}{\binom{V+W-1}{W}} = 1.
\end{aligned}
\tag{B.5}
$$

The position above induces the definition of the weighted surprise as $\mathscr{W}=\sum_{w_\bullet\geq w_\bullet^*}f(w_\bullet)$. The generalization of the formalism above to detect 'bimodular' structures on weighted networks is straightforward. It is enough to consider the multivariate (or multinomial) version of the negative hypergeometric distribution, i.e.

The asymptotic expression of $\mathscr{W}$ and $\mathscr{W}_{/\!/}$ can be derived by applying the Stirling approximation $n!\simeq\sqrt{2\pi n}\left(\frac{n}{e}\right)^n$ to the binomial coefficients appearing into the definition of the versions of the surprise described above, leading to expressions like

$$\begin{aligned}\binom{V+W-1}{W} &= \frac{(V+W-1)!}{W!(V-1)!}\\ &\simeq \frac{\sqrt{2\pi(V+W-1)}(V+W-1)^{V+W-1}e^{-(V+W-1)}}{\sqrt{2\pi W}W^W e^{-W}\sqrt{2\pi(V-1)}(V-1)^{V-1}e^{-(V-1)}}\\ &= \frac{1}{\sqrt{2\pi(V-1)\left(\frac{W}{V+W-1}\right)}}\cdot\frac{(V+W-1)^{V+W-1}}{W^W(V-1)^{V-1}}\\ &\simeq \frac{1}{\sqrt{2\pi Vq}}\cdot\frac{1}{\left(\frac{W}{V+W-1}\right)^W\left(\frac{V-1}{V+W-1}\right)^{V-1}}\\ &\simeq \frac{1}{\sqrt{2\pi Vq}}\cdot\frac{1}{q^W(1-q)^{V-1}}\\ &= \frac{1}{\sqrt{2\pi\mu}\cdot\mathrm{Geo}(V,W,q)} \end{aligned} \tag{B.6}$$

where $\mathrm{Geo}(x,y,z)=z^y(1-z)^x$ is a geometric probability mass function, $z=\frac{y}{x+y}$ and $q\simeq\frac{W}{V+W}$.

The time, in seconds, to find the corresponding optimal partitions is reported in tables 9 and 10: overall, denser networks require more time.

Network	**# nodes** (N)	**# links** (L)	**Connectance**	**Binary** (s)	**Weighted** (s)	**Enhanced** (s)
Madrid terrorists	64	486	0.24	0.05	0.05	0.07
Star Wars	110	398	0.07	0.13	0.14	0.24
Australian college	217	2672	0.11	0.34	0.30	0.38

Table 9: Time, in seconds, employed by $\mathscr{S}$, $\mathscr{W}$ and $\mathscr{E}$ to find the corresponding optimal partition.

$$
\begin{aligned}
f(w_\bullet, w_\circ) &= \text{MNH}(w_\bullet, w_\circ | V+W, W, V_\bullet, V_\circ)\\
&= \frac{\prod_{i=\bullet,\circ,\top}\binom{V_i+w_i-1}{w_i}}{\binom{V+W-1}{W}}\\
&= \frac{\binom{V_\bullet+w_\bullet-1}{w_\bullet}\binom{V_\circ+w_\circ-1}{w_\circ}\binom{V_\top+w_\top-1}{w_\top}}{\binom{V+W-1}{W}}\\
&= \frac{\binom{V_\bullet+w_\bullet-1}{w_\bullet}\binom{V_\circ+w_\circ-1}{w_\circ}\binom{V-(V_\bullet+V_\circ)+W-(w_\bullet+w_\circ)-1}{W-(w_\bullet+w_\circ)}}{\binom{V+W-1}{W}}
\end{aligned}
\tag{B.7}
$$

whose correctedness can be checked upon calculating its normalization, by applying the identity $\sum_{k=0}^{r}\binom{k+m}{k}\binom{n-m-k}{r-k} = \binom{n+1}{r}$ twice. The position above, in turn, induces the definition of the weighted bimodular surprise, as $\mathscr{W}_{/\!/} = \sum_{w_\bullet \geq w_\bullet^*}\sum_{w_\circ \geq w_\circ^*} f(w_\bullet, w_\circ)$.

B.3 Enhanced detection of mesoscale structures

The enhanced hypergeometric distribution is a negative hypergeometric distribution 'conditional' to the existence of connections within communities, i.e. $l_\bullet > 0$. It reads

$$
\text{EH}(l_\bullet, w_\bullet) = \begin{cases} \frac{\binom{V_\bullet}{l_\bullet}\binom{V_\circ}{l_\circ}}{\binom{V}{L}} \cdot \frac{\binom{w_\bullet-1}{w_\bullet-l_\bullet}\binom{w_\circ-1}{w_\circ-l_\circ}}{\binom{W-1}{L-1}}, & 0 < l_\bullet < L \\ \frac{\binom{V_\bullet}{L}}{\binom{V}{L}} \cdot \frac{\binom{w_\bullet-1}{w_\bullet-L}}{\binom{W}{L}}, & l_\bullet = L \end{cases}
\tag{B.8}
$$

the first expression being valid when $l_\bullet < L$ and the second expression being valid in the 'extreme' case $l_\bullet = L$ (i.e. whenever the total weight

Network	**# nodes** (N)	**# links** (L)	**Connectance**	**Binary** (s)	**Weighted** (s)	**Enhanced** (s)
'Les Miserables'	77	254	0.09	0.22	0.31	0.48
e-MID 2009	134	4609	0.26	1.42	0.80	4.12
WTW 2000	187	20105	0.58	12.16	12.05	21.23

Table 10: Time, in seconds, employed by $\mathscr{S}_{/\!/}$, $\mathscr{W}_{/\!/}$ and $\mathscr{E}_{/\!/}$ to find the corresponding optimal partition.

has to be redistributed on top of links which are all put inside the communities). Normalization is ensured by the fact that

$$\sum_{l_\bullet=1}^{L-1}\frac{\binom{V_\bullet}{l_\bullet}\binom{V-V_\bullet}{L-l_\bullet}}{\binom{V}{L}}\cdot\sum_{w_\bullet-l_\bullet=0}^{W-L}\frac{\binom{(w_\bullet-l_\bullet)+(l_\bullet-1)}{w_\bullet-l_\bullet}\binom{(W-2)-(l_\bullet-1)-(w_\bullet-l_\bullet)}{(W-L)-(w_\bullet-l_\bullet)}}{\binom{W-1}{L-1}}$$
$$+\frac{\binom{V_\bullet}{L}}{\binom{V}{L}}\cdot\sum_{w_\bullet=L}^{W}\frac{\binom{w_\bullet-1}{w_\bullet-L}}{\binom{W}{L}}=\sum_{l_\bullet=1}^{L}\frac{\binom{V_\bullet}{l_\bullet}\binom{V-V_\bullet}{L-l_\bullet}}{\binom{V}{L}}=1-\frac{\binom{V_\circ}{L}}{\binom{V}{L}} \tag{B.9}$$

where we have used the relationships $\sum_{k=0}^{r}\binom{k+m}{k}\binom{n-m-k}{r-k}=\binom{n+1}{r}$, $\sum_{k=r}^{n}\binom{k}{r}=\sum_{k=r}^{n}\binom{k}{k-r}=\binom{n+1}{r+1}$ and $\sum_{k=0}^{r}\binom{m}{k}\binom{n-m}{r-k}=\binom{n}{r}$ and assumed that $L<V_\bullet$. For what concerns the expected value, we have that

$$\langle l_\bullet\rangle=\sum_{l_\bullet=1}^{L-1}l_\bullet\cdot\frac{\binom{V_\bullet}{l_\bullet}\binom{V-V_\bullet}{L-l_\bullet}}{\binom{V}{L}}\cdot\sum_{w_\bullet-l_\bullet=0}^{W-L}\frac{\binom{(w_\bullet-l_\bullet)+(l_\bullet-1)}{w_\bullet-l_\bullet}\binom{(W-2)-(l_\bullet-1)-(w_\bullet-l_\bullet)}{(W-L)-(w_\bullet-l_\bullet)}}{\binom{W-1}{L-1}}$$
$$+L\cdot\frac{\binom{V_\bullet}{L}}{\binom{V}{L}}\cdot\sum_{w_\bullet=L}^{W}\frac{\binom{w_\bullet-1}{w_\bullet-L}}{\binom{W}{L}}=\sum_{l_\bullet=1}^{L}l_\bullet\cdot\frac{\binom{V_\bullet}{l_\bullet}\binom{V-V_\bullet}{L-l_\bullet}}{\binom{V}{L}}=L\frac{V_\bullet}{V} \tag{B.10}$$

as it should be: in fact, the expected value of (the number of) connections should not depend on the weights put on the connections. Moreover,

$$
\begin{aligned}
\langle w_\bullet - l_\bullet \rangle =& \sum_{l_\bullet=1}^{L-1} \sum_{w_\bullet - l_\bullet=0}^{W-L} (w_\bullet - l_\bullet) \cdot \frac{\binom{V_\bullet}{l_\bullet}\binom{V-V_\bullet}{L-l_\bullet}}{\binom{V}{L}} \\
&\cdot \frac{\binom{(w_\bullet-l_\bullet)+(l_\bullet-1)}{w_\bullet-l_\bullet}\binom{(W-2)-(l_\bullet-1)-(w_\bullet-l_\bullet)}{(W-L)-(w_\bullet-l_\bullet)}}{\binom{W-1}{L-1}} \\
&+(W-L) \cdot \frac{\binom{V_\bullet}{L}}{\binom{V}{L}} \cdot \sum_{w_\bullet=L}^{W} \frac{\binom{w_\bullet-1}{w_\bullet-L}}{\binom{W}{L}} \\
=& \sum_{l_\bullet=1}^{L-1} \frac{\binom{V_\bullet}{l_\bullet}\binom{V-V_\bullet}{L-l_\bullet}}{\binom{V}{L}} \cdot \sum_{w_\bullet-l_\bullet=0}^{W-L} (w_\bullet - l_\bullet) \\
&\cdot \frac{\binom{(w_\bullet-l_\bullet)+(l_\bullet-1)}{w_\bullet-l_\bullet}\binom{(W-2)-(l_\bullet-1)-(w_\bullet-l_\bullet)}{(W-L)-(w_\bullet-l_\bullet)}}{\binom{W-1}{L-1}} + (W-L) \cdot \frac{\binom{V_\bullet}{L}}{\binom{V}{L}} \\
=& \sum_{l_\bullet=1}^{L-1} \frac{\binom{V_\bullet}{l_\bullet}\binom{V-V_\bullet}{L-l_\bullet}}{\binom{V}{L}} \cdot l_\bullet \cdot \frac{W-L}{L} + (W-L) \cdot \frac{\binom{V_\bullet}{L}}{\binom{V}{L}} \\
=& L\left(\frac{V_\bullet}{V} - \frac{\binom{V_\bullet}{L}}{\binom{V}{L}}\right)\left(\frac{W-L}{L}\right) + (W-L) \cdot \frac{\binom{V_\bullet}{L}}{\binom{V}{L}} \\
=& (W-L)\frac{V_\bullet}{V}. \qquad \text{(B.11)}
\end{aligned}
$$

The asymptotic expression of $\mathscr{E}$ can be derived by applying the Stirling approximation $n! \simeq \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$ to the binomial coefficients appearing into the definition of the enhanced surprise, leading to expressions like

$$
\begin{aligned}
\binom{V}{L}\binom{W-1}{W-L} =& \frac{V!}{L!(V-L)!} \cdot \frac{(W-1)!}{(W-L)!(L-1)!} \\
\propto& \frac{V^V}{L^L(V-L)^{V-L}} \cdot \frac{(W-1)^{W-1}}{(W-L)^{W-L}(L-1)^{L-1}} \\
\simeq& \frac{V^V}{L^L(V-L)^{V-L}} \cdot \frac{W^W}{(W-L)^{W-L}L^L} \\
=& \frac{1}{\left(\frac{L}{V-L}\right)^L \left(1-\frac{L}{V}\right)^V} \cdot \frac{1}{\left(1-\frac{L}{W}\right)^W \left(\frac{L}{W-L}\right)^L} \\
=& \frac{1}{\left(\frac{L}{V-L}\right)^L \left(1-\frac{L}{V}\right)^V} \cdot \frac{1}{\left(1-\frac{L}{W}\right)^{W-L} \left(\frac{L}{W}\right)^L} \\
=& \frac{1}{p^L(1-p)^{V-L}} \cdot \frac{1}{q^{W-L}(1-q)^L} \\
=& \mathrm{BF}[V, W, L, p, q] \\
=& \mathrm{Ber}(V, L, p) \cdot \mathrm{Geo}(L, W-L, q) \qquad \text{(B.12)}
\end{aligned}
$$

with a clear meaning of the symbols. A graphical representation of the enhanced hypergeometric distribution is provided in fig. 33.

Let us now briefly consider the distribution inducing the definition of the enhanced bimodular surprise, i.e. the 'multivariate' enhanced hypergeometric distribution. As before, it is 'conditional' to the existence of connections within either modules one is looking at (i.e. either $l_\bullet > 0$ or $l_\circ > 0$) and reads

$$
\mathrm{MEH}(l_\bullet, l_\circ, w_\bullet, w_\circ) = \begin{cases} \frac{\binom{V_\bullet}{l_\bullet}\binom{V_\circ}{l_\circ}\binom{V_\top}{l_\top}}{\binom{V}{L}} \cdot \frac{\binom{w_\bullet-1}{w_\bullet-l_\bullet}\binom{w_\circ-1}{w_\circ-l_\circ}\binom{w_\top-1}{w_\top-l_\top}}{\binom{W-1}{W-L}}, & 0 < l_\bullet, l_\circ < L \\ \frac{\binom{V_\bullet}{L}}{\binom{V}{L}} \cdot \frac{\binom{w_\bullet-1}{w_\bullet-L}}{\binom{W}{L}}, & l_\bullet = L,\ l_\circ = 0 \\ \frac{\binom{V_\circ}{l_\circ}}{\binom{V}{L}} \cdot \frac{\binom{w_\circ-1}{w_\circ-L}}{\binom{W}{L}}, & l_\bullet = 0,\ l_\circ = L \end{cases} \qquad \text{(B.13)}
$$

with a clear meaning of the symbols. Notice that the cases $l_\bullet = 0$, $0 < l_\circ < L$ and $l_\circ = 0$, $0 < l_\bullet < L$ can be easily recovered from the first row of the definition above.

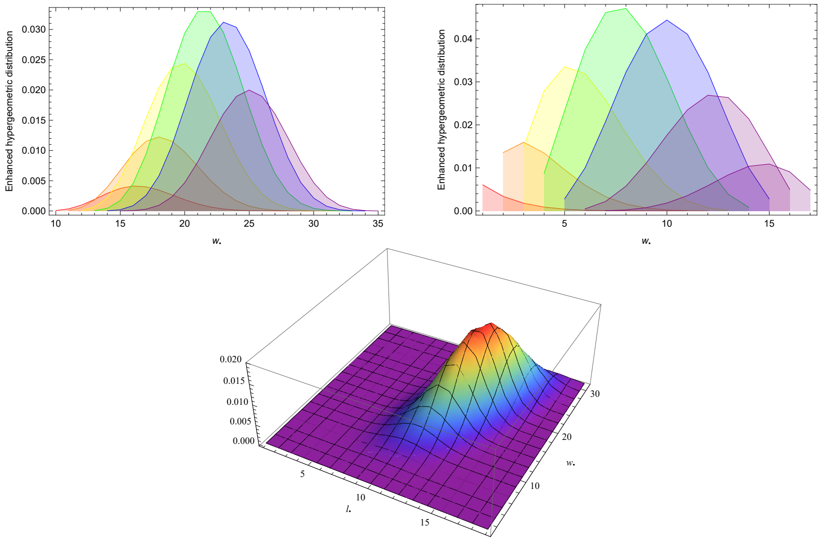


Figure 33: Examples of the enhanced hypergeometric distribution $\text{EH}(l_\bullet, w_\bullet | V, V_\bullet, L, W)$. Top-left panel: instances characterized by the values $V = 45$, $V_\bullet = 20$, $L = 30$, $W = 50$ and $l_\bullet = 10, 11, 12, 13, 14, 15$. Top-right panel: instances characterized by the values $V = 45$, $V_\bullet = 20$, $L = 10$, $W = 20$ and $l_\bullet = 1, 2, 3, 4, 5, 6, 7$. Bottom panel: full view of the enhanced hypergeometric distribution with parameters $V = 80$, $V_\bullet = 50$, $L = 20$, $W = 50$ and $0 < l_\bullet < L$.

B.4 Pseudocodes for surprise optimization

Let us now provide a description of the algorithms implemented in the 'SurpriseMeMore' Python package: to this aim, we will also show the corresponding pseudocodes.

Since an exhaustive exploration of the space of all possible partitions of a network is not feasible (especially when dealing with large graphs), one often proceeds heuristically, either in an *agglomerative* fashion or fixing the number of clusters to be detected *a priori*.

B.4.1 Binary and weighted community detection

The algorithm for community detection on binary networks is called PACO (PArtitioning Cost Optimization) and its pseudocode can be found in [37]. The algorithm implemented there is agglomerative in nature: in the initial configuration of the algorithm, in fact, nodes are treated as singletons (hence, there are as many communities as nodes). Then, edges are sorted according to the value of their Jaccard index, in a decreasing order: as the Jaccard index of an edge is proportional to the number of common neighbours of the two connected nodes, a large value of it likely points out that the two nodes must be assigned to the same module. For further details about PACO, see [37].

Here, we adopt a similar strategy to the one implemented by PACO in the binary case, with two main differences. The first one concerns the edge-sorting step: edges, in fact, are no longer sorted according to the Jaccard index but randomly ordered; this strategy avoids getting stuck in local minima more effectively. The second one concerns the possibility of letting entire communities merge (instead of moving single nodes from one to another) with a certain probability p_{mix}.

```
1: function CALCULATEANDUPDATESURPRISE($C, C'$)
2:     $S \leftarrow calculateSurprise(C)$
3:     $S' \leftarrow calculateSurprise(C')$
4:     if $S' < S$ then
5:         $C \leftarrow C'$
6:         $S \leftarrow S'$
7:     end if
8: return $C$
9: end function
10:
11: $C \leftarrow$ array of length $N$, randomly initialized with $N$ different integers, i.e. $1, 2 \dots N$;
12: $E \leftarrow$ randomly sorted edges;
13: for edge $(u, v) \in E$ do
```

```
14:     if $C[u] \neq C[v]$ then
15:         $C' \leftarrow C$
16:         if $0 \leq uniformDistribution(0,1) \leq p_{mix}$ then
17:             for node $t \in C[u]$ do
18:                 $C'[t] \leftarrow C[v]$
19:             end for
20:             $C \leftarrow$CALCULATEANDUPDATESURPRISE($C, C'$)
21:         end if
22:         if $p_{mix} \leq uniformDistribution(0,1) \leq p_{single}$ then
23:             $C'[u] \leftarrow C[v]$
24:             $C \leftarrow$CALCULATEANDUPDATESURPRISE($C, C'$)
25:         end if
26:         if $p_{single} \leq uniformDistribution(0,1) \leq 1$ then
27:             $C'[v] \leftarrow C[u]$
28:             $C \leftarrow$CALCULATEANDUPDATESURPRISE($C, C'$)
29:         end if
30:     end if
31:     $\Rightarrow$ consider each node and swap its membership with the one of
    each first neighbour of it; accept the move if $\mathscr{S}$, $\mathscr{W}$ or $\mathscr{E}$ decreases;
32: end for
33: $\Rightarrow$ repeat the for-loop to improve the chance of finding the optimal
    partition
```

Remarkably, it is also possible to define a version of the surprise-based community-detection algorithm, where the number of clusters to be detected is fixed a priori:

```
1: function CALCULATEANDUPDATESURPRISE($C, C'$)
2:     $S \leftarrow calculateSurprise(C)$
3:     $S' \leftarrow calculateSurprise(C')$
4:     if $S' < S$ then
5:         $C \leftarrow C'$
6:         $S \leftarrow S'$
7:     end if
```

```
    return $C$
end function

$C \leftarrow$ array of length $N$, randomly initialized with $n_{clust}$ different integers, i.e. $1, 2 \ldots n_{clust}$;
$E \leftarrow$ randomly sorted edges;
for edge $(u, v) \in E$ do
    $C' \leftarrow C$
    if $C'[u] \neq C[v]$ then
        $C'[u] \leftarrow C[v]$
        $C[v] \leftarrow C'[u]$
        $C \leftarrow$ CALCULATEANDUPDATESURPRISE($C, C'$)
    else
        $C'[u] \leftarrow randomInteger(1, n_{clust})$
        $C \leftarrow$ CALCULATEANDUPDATESURPRISE($C, C'$)
        $C''[v] \leftarrow randomInteger(1, n_{clust})$
        $C \leftarrow$ CALCULATEANDUPDATESURPRISE($C, C''$)
    end if
    $\Rightarrow$ consider each node and swap its membership with the one of each first neighbour of it; accept the move if $\mathscr{S}$, $\mathscr{W}$ or $\mathscr{E}$ decreases;
end for
$\Rightarrow$ repeat the for-loop to improve the chance of finding the optimal partition
```

B.4.2 Binary and weighted bimodular structures detection

In this case, the algorithm we implement fixes the number of clusters to be detected a priori, since each node of the network can belong to only one out of two possible subsets (e.g. the core and the periphery) - a circumstance that is reflected into the initialization of the algorithm, i.e. a vector whose entries are randomly chosen to be either 0 or 1. A second difference with respect to the community detection case concerns the criterion according to which edges are sorted (for purely bipartite graphs, in fact, the Jaccard index is zero for all edges, as nodes connected by an edge always lie on different layers): in our modified version of the PACO algorithm, when binary graphs are considered, we sort links according to their eigenvector centrality, which has been shown to proxy 'coreness' to quite a good extent. As a final step, we consider a number of random re-assignments of nodes with the aim of preventing the possibility of getting stuck in local minima (a random move consists of selecting 3 nodes belonging to the same group and evaluating if assigning them to the other group would further minimize surprise).

The edge-sorting strategy described above is not the best one in the weighted case: in this case, in fact, randomly-ordered edges are to be preferred, as they avoid getting stuck in local minima more effectively.

```
1: function CalculateAndUpdateSurprise($C, C'$)
2:     $S \leftarrow calculateSurprise(C)$
3:     $S' \leftarrow calculateSurprise(C')$
4:     if $S' < S$ then
5:         $C \leftarrow C'$
6:         $S \leftarrow S'$
7:     end if
8: return $C$
9: end function
10:
11: $C \leftarrow$ array of length $N$, randomly initialized with binary entries, i.e. $0, 1$;
12: $E \leftarrow$ sorted edges (either via the eigenvector centrality, in case of binary networks, or randomly, in case of weighted networks);
13: for edge $(u, v) \in E$ do
14:     $C' \leftarrow C$
15:     if $C'[u] \neq C[v]$ then
16:         $C'[u] \leftarrow C[v]$
17:         $C \leftarrow$CalculateAndUpdateSurprise($C, C'$)
18:     else
19:         $C'[u] \leftarrow 1 - C[v]$
20:         $C \leftarrow$CalculateAndUpdateSurprise($C, C'$)
```

```
21:     end if
22:     ⇒ randomly switch the membership of 3 nodes in the same group
        and accept the move if $\mathscr{S}_{/\!/}$, $\mathscr{W}_{/\!/}$ or $\mathscr{E}_{/\!/}$ decreases;
23: end for
24: ⇒ repeat the for-loop to improve the chance of finding the optimal
    partition
```

Bibliography

[1] Joris Toonders. *Data is the new oil of digital economy.* 2014. URL: https://www.wired.com/insights/2014/07/data-new-oil-digital-economy/ (visited on 09/27/2021).

[2] Kiran Bhageshpur. *Data is the new oil and that's a good thing.* 2019. URL: https://www.forbes.com/sites/forbestechcouncil/2019/11/15/data-is-the-new-oil-and-thats-a-good-thing/ (visited on 09/27/2021).

[3] The Economist. *The world's most valuable resource is no longer oil, but data.* 2017. URL: https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data (visited on 09/27/2021).

[4] Mark Newman. *Networks*. Oxford university press, 2018.

[5] Vittoria Colizza et al. "The role of the airline transportation network in the prediction and predictability of global epidemics". In: *Proceedings of the National Academy of Sciences* 103.7 (2006), pp. 2015–2020.

[6] Alain Barrat, Marc Barthelemy, and Alessandro Vespignani. *Dynamical processes on complex networks*. Cambridge university press, 2008.

[7] Romualdo Pastor-Satorras et al. "Epidemic processes in complex networks". In: *Reviews of modern physics* 87.3 (2015), p. 925.

[8] Claudio Castellano, Santo Fortunato, and Vittorio Loreto. "Statistical physics of social dynamics". In: *Reviews of modern physics* 81.2 (2009), p. 591.

[9] Tiziano Squartini, Iman Van Lelyveld, and Diego Garlaschelli. “Early-warning signals of topological collapse in interbank networks”. In: *Scientific reports* 3.1 (2013), pp. 1–9.

[10] Paolo Crucitti, Vito Latora, and Massimo Marchiori. “Model for cascading failures in complex networks”. In: *Physical Review E* 69.4 (2004), p. 045104.

[11] Ryan Kinney et al. “Modeling cascading failures in the North American power grid”. In: *The European Physical Journal B-Condensed Matter and Complex Systems* 46.1 (2005), pp. 101–107.

[12] Adilson E Motter and Ying-Cheng Lai. “Cascade-based attacks on complex networks”. In: *Physical Review E* 66.6 (2002), p. 065102.

[13] Adilson E Motter. “Cascade control and defense in complex networks”. In: *Physical review letters* 93.9 (2004), p. 098701.

[14] Giulio Cimini et al. “The statistical physics of real-world networks”. In: *Nature Reviews Physics* 1.1 (2019), pp. 58–71.

[15] Sergei Maslov and Kim Sneppen. “Specificity and stability in topology of protein networks”. In: *Science* 296.5569 (2002), pp. 910–913.

[16] ACC Coolen, Andrea De Martino, and Alessia Annibale. “Constrained Markovian dynamics of random graphs”. In: *Journal of Statistical Physics* 136.6 (2009), pp. 1035–1067.

[17] ES Roberts and ACC Coolen. “Unbiased degree-preserving randomization of directed binary networks”. In: *Physical Review E* 85.4 (2012), p. 046103.

[18] Yael Artzy-Randrup and Lewi Stone. “Generating uniformly distributed random networks”. In: *Physical Review E* 72.5 (2005), p. 056708.

[19] Charo I Del Genio et al. “Efficient and exact sampling of simple graphs with given arbitrary degree sequence”. In: *PloS one* 5.4 (2010), e10012.

[20] Hyunju Kim et al. “Constructing and sampling directed graphs with given degree sequences”. In: *New Journal of Physics* 14.2 (2012), p. 023012.

[21] Joseph Blitzstein and Persi Diaconis. “A sequential importance sampling algorithm for generating random graphs with prescribed degrees”. In: *Internet mathematics* 6.4 (2011), pp. 489–522.

[22] Tiziano Squartini and Diego Garlaschelli. “Analytical maximum-likelihood method to detect patterns in real networks”. In: *New Journal of Physics* 13.8 (2011), p. 083001.

[23] Juyong Park and Mark EJ Newman. “Statistical mechanics of networks”. In: *Physical Review E* 70.6 (2004), p. 066117.

[24] Ginestra Bianconi. “The entropy of randomized network ensembles”. In: *EPL (Europhysics Letters)* 81.2 (2007), p. 28005.

[25] Agata Fronczak, Piotr Fronczak, and Janusz A Hołyst. “Fluctuation-dissipation relations in complex networks”. In: *Physical Review E* 73.1 (2006), p. 016108.

[26] Andrea Gabrielli et al. “Grand canonical ensemble of weighted networks”. In: *Physical Review E* 99.3 (2019), p. 030301.

[27] P. Erdös and A. Rényi. “On Random Graphs I”. In: *Publicationes Mathematicae Debrecen* 6 (1959), p. 290.

[28] Edgar N Gilbert. “Random graphs”. In: *The Annals of Mathematical Statistics* 30.4 (1959), pp. 1141–1144.

[29] Paul W Holland and Samuel Leinhardt. “An exponential family of probability distributions for directed graphs”. In: *Journal of the american Statistical association* 76.373 (1981), pp. 33–50.

[30] Ove Frank and David Strauss. “Markov graphs”. In: *Journal of the american Statistical association* 81.395 (1986), pp. 832–842.

[31] Nicolò Vallarano et al. “Fast and scalable likelihood maximization for Exponential Random Graph Models with local constraints”. In: *Scientific Reports* 11.1 (2021), pp. 1–33.

[32] Andrea Lancichinetti and Santo Fortunato. “Community detection algorithms: a comparative analysis”. In: *Physical review E* 80.5 (2009), p. 056117.

[33] Rodrigo Aldecoa and Ignacio Marin. “SurpriseMe: an integrated tool for network community structure characterization using Surprise maximization”. In: *Bioinformatics* 30.7 (2014), pp. 1041–1042.

[34] Vincent D Blondel et al. “Fast unfolding of communities in large networks”. In: *Journal of statistical mechanics: theory and experiment* 2008.10 (2008), P10008.

[35] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. “Finding community structure in very large networks”. In: *Physical review E* 70.6 (2004), p. 066111.

[36] Martin Rosvall and Carl T Bergstrom. “Maps of random walks on complex networks reveal community structure”. In: *Proceedings of the national academy of sciences* 105.4 (2008), pp. 1118–1123.

[37] Carlo Nicolini and Angelo Bifone. “Modular structure of brain functional networks: breaking the resolution limit by Surprise”. In: *Scientific reports* 6.1 (2016), pp. 1–13.

[38] J van Lidth de Jeude, Guido Caldarelli, and Tiziano Squartini. “Detecting core-periphery structures by surprise”. In: *EPL (Europhysics Letters)* 125.6 (2019), p. 68001.

[39] Zhao Yang, René Algesheimer, and Claudio J Tessone. “A comparative analysis of community detection algorithms on artificial networks”. In: *Scientific reports* 6.1 (2016), pp. 1–18.

[40] Michel Plantié and Michel Crampes. “Survey on social community detection”. In: *Social media retrieval*. Springer, 2013, pp. 65–85.

[41] Rodrigo Aldecoa and Ignacio Marin. “Surprise maximization reveals the community structure of complex networks”. In: *Scientific reports* 3.1 (2013), pp. 1–9.

[42] Emiliano Marchese, Guido Caldarelli, and Tiziano Squartini. *Detecting mesoscale structures by surprise*. 2021. arXiv: `2106.05055 [physics.soc-ph]`.

[43] Agata Fronczak. *Exponential random graph models*. 2012. arXiv: `1210.7828 [physics.soc-ph]`.

[44] Edwin T Jaynes. “Information theory and statistical mechanics”. In: *Physical review* 106.4 (1957), p. 620.

[45] Diego Garlaschelli and Maria I Loffredo. “Maximum likelihood: extracting unbiased information from complex networks”. In: *Physical Review E* 78.1 (2008), p. 015101.

[46] Jorge Nocedal and Stephen J Wright. “Conjugate gradient methods”. In: *Numerical optimization* (2006), pp. 101–134.

[47] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[48] Navid Dianati. *A maximum entropy approach to separating noise from signal in bimodal affiliation networks*. 2016. arXiv: `1607.01735 [physics.soc`

[49] Nicoló Vallarano, Claudio J Tessone, and Tiziano Squartini. "Bitcoin Transaction Networks: an overview of recent results". In: *Frontiers in Physics* 8 (2020), p. 286.

[50] Fan Chung and Linyuan Lu. "Connected components in random graphs with given expected degree sequences". In: *Annals of combinatorics* 6.2 (2002), pp. 125–145.

[51] K. Oshio et al. In: *Tech. Rep. of CCeP, Keio Future* 3 (2003).

[52] Vittoria Colizza, Romualdo Pastor-Satorras, and Alessandro Vespignani. "Reaction–diffusion processes and metapopulation models in heterogeneous networks". In: *Nature Physics* 3.4 (2007), pp. 276–282.

[53] *Database of Interacting Proteins*. URL: `http://dip.doe-mbi.ucla.edu/dip/Main.cgi`.

[54] Vittoria Colizza et al. "Detecting rich-club ordering in complex networks". In: *Nature physics* 2.2 (2006), pp. 110–115.

[55] Jian-Hong Lin et al. "Lightning Network: a second path towards centralisation of the Bitcoin economy". In: *New Journal of Physics* 22.8 (2020), p. 083022.

[56] Joel C Miller and Aric Hagberg. "Efficient generation of networks with given expected degrees". In: *International Workshop on Algorithms and Models for the Web-Graph*. Springer. 2011, pp. 115–126.

[57] Rossana Mastrandrea et al. "Reconstructing the world trade multiplex: the role of intensive and extensive biases". In: *Physical Review E* 90.6 (2014), p. 062804.

[58] Diego Garlaschelli and Maria I Loffredo. "Generalized bose-fermi statistics and structural correlations in weighted networks". In: *Physical review letters* 102.3 (2009), p. 038701.

[59] Rossana Mastrandrea et al. "Reconstructing the world trade multiplex: the role of intensive and extensive biases". In: *Physical Review E* 90.6 (2014), p. 062804.

[60] Kristian Skrede Gleditsch. "Expanded trade and GDP data". In: *Journal of Conflict Resolution* 46.5 (2002), pp. 712–724.

[61] Giulia Iori et al. "A network analysis of the Italian overnight money market". In: *Journal of Economic Dynamics and Control* 32.1 (2008), pp. 259–278. URL: `https://EconPapers.repec.org/RePEc:eee:dyncon:v:32:y:2008:i:1:p:259-278`.

[62] Federica Parisi, Tiziano Squartini, and Diego Garlaschelli. “A faster horse on a safer trail: generalized inference for the efficient reconstruction of weighted networks”. In: *New Journal of Physics* 22.5 (2020), p. 053053.

[63] Giulio Cimini et al. “Estimating topological properties of weighted networks from limited information”. In: *Physical Review E* 92.4 (2015), p. 040802.

[64] Giulio Cimini et al. “Systemic risk analysis on reconstructed economic and financial networks”. In: *Scientific reports* 5.1 (2015), pp. 1–12.

[65] *‘NEMTROPY’ module*. URL: `https://github.com/nicoloval/NEMtropy`.

[66] Tiziano Squartini, Rossana Mastrandrea, and Diego Garlaschelli. “Unbiased sampling of network ensembles”. In: *New Journal of Physics* 17.2 (2015), p. 023052.

[67] Santo Fortunato and Darko Hric. “Community detection in networks: A user guide”. In: *Physics reports* 659 (2016), pp. 1–44.

[68] Bisma S Khan and Muaz A Niazi. “Network community detection: A review and visual survey”. In: *arXiv preprint arXiv:1708.00977* (2017).

[69] Stephen P Borgatti and Martin G Everett. “Models of core/periphery structures”. In: *Social networks* 21.4 (2000), pp. 375–395.

[70] Ben Craig and Goetz Von Peter. “Interbank tiering and money center banks”. In: *Journal of Financial Intermediation* 23.3 (2014), pp. 322–347.

[71] Iman Van Lelyveld et al. “Finding the core: Network structure in interbank markets”. In: *Journal of Banking & Finance* 49 (2014), pp. 27–40.

[72] Duc Thi Luu et al. “Collateral Unchained: Rehypothecation networks, concentration and systemic effects”. In: *Journal of Financial Stability* 52 (2021), p. 100811.

[73] Santo Fortunato. “Community detection in graphs”. In: *Physics reports* 486.3-5 (2010), pp. 75–174.

[74] Giulio Cimini, Rossana Mastrandrea, and Tiziano Squartini. “Reconstructing networks”. In: *arXiv preprint arXiv:2012.02677* (2020).

[75] Mark EJ Newman and Michelle Girvan. “Finding and evaluating community structure in networks”. In: *Physical review E* 69.2 (2004), p. 026113.

[76] Michael J Barber. “Modularity and community detection in bipartite networks”. In: *Physical Review E* 76.6 (2007), p. 066102.

[77] Vincent A Traag and Jeroen Bruggeman. “Community detection in networks with positive and negative links”. In: *Physical Review E* 80.3 (2009), p. 036115.

[78] Mel MacMahon and Diego Garlaschelli. “Community Detection for Correlation Matrices”. In: *Phys. Rev. X* 5 (2 2015), p. 021006. DOI: `10.1103/PhysRevX.5.021006`. URL: `https://link.aps.org/doi/10.1103/PhysRevX.5.021006`.

[79] Brian Ball, Brian Karrer, and Mark EJ Newman. “Efficient and principled method for detecting communities in networks”. In: *Physical Review E* 84.3 (2011), p. 036103.

[80] Roger Guimerà and Marta Sales-Pardo. “Missing and spurious interactions and the reconstruction of complex networks”. In: *Proceedings of the National Academy of Sciences* 106.52 (2009), pp. 22073–22078.

[81] Matthew B Hastings. “Community detection as an inference problem”. In: *Physical Review E* 74.3 (2006), p. 035102.

[82] Brian Karrer and Mark EJ Newman. “Stochastic blockmodels and community structure in networks”. In: *Physical review E* 83.1 (2011), p. 016107.

[83] Elizabeth A Leicht and Mark EJ Newman. “Community structure in directed networks”. In: *Physical review letters* 100.11 (2008), p. 118703.

[84] Tiago P Peixoto. “Hierarchical block structures and high-resolution model selection in large networks”. In: *Physical Review X* 4.1 (2014), p. 011047.

[85] Peter D Grunwald, In Jae Myung, and Mark A Pitt. *Advances in minimum description length: theory and applications (Neural Information Processing)*. 2005.

[86] Tiago P Peixoto. “Parsimonious module inference in large networks”. In: *Physical review letters* 110.14 (2013), p. 148701.

[87] Tiago P Peixoto. “Bayesian stochastic blockmodeling”. In: *Advances in network clustering and blockmodeling* (2019), pp. 289–332.

[88] Michele Tumminello et al. “Statistically validated networks in bipartite complex systems”. In: *PloS one* 6.3 (2011), e17994.

[89] Christian Bongiorno et al. “Core of communities in bipartite networks”. In: *Physical Review E* 96.2 (2017), p. 022321.

[90] Federico Musciotto, Federico Battiston, and Rosario N Mantegna. “Detecting informative higher-order interactions in statistically validated hypergraphs”. In: *arXiv preprint arXiv:2103.16484* (2021).

[91] Salvatore Miccichè and Rosario N Mantegna. “A primer on statistically validated networks”. In: *Computational Social Science and Complex Systems* 203 (2019), p. 91.

[92] Yawen Jiang, Caiyan Jia, and Jian Yu. “An efficient community detection algorithm using greedy surprise maximization”. In: *Journal of Physics A: Mathematical and Theoretical* 47.16 (2014), p. 165101.

[93] Javier Del Ser et al. “Community detection in graphs based on surprise maximization using firefly heuristics”. In: *2016 IEEE congress on evolutionary computation (CEC)*. IEEE. 2016, pp. 2233–2239.

[94] Yan-Ni Tang et al. “An effective algorithm for optimizing surprise in network community detection”. In: *IEEE Access* 7 (2019), pp. 148814–148827.

[95] Daniel Gamermann and José Antônio Pellizaro. “An algorithm for network community structure determination by surprise”. In: *arXiv preprint arXiv:2012.13780* (2020).

[96] Sadamori Kojaku and Naoki Masuda. “A generalised significance test for individual communities in networks”. In: *Scientific reports* 8.1 (2018), pp. 1–10.

[97] Vincent A Traag, Rodrigo Aldecoa, and J-C Delvenne. “Detecting communities using asymptotical surprise”. In: *Physical review e* 92.2 (2015), p. 022816.

[98] Xiao Zhang, Travis Martin, and Mark EJ Newman. “Identification of core-periphery structure in networks”. In: *Physical Review E* 91.3 (2015), p. 032803.

[99] Paolo Barucca and Fabrizio Lillo. “Disentangling bipartite and core-periphery structure in financial networks”. In: *Chaos, Solitons & Fractals* 88 (2016), pp. 244–253.

[100] Sadamori Kojaku and Naoki Masuda. "Core-periphery structure requires something else in the network". In: *New Journal of physics* 20.4 (2018), p. 043012.

[101] Petter Holme et al. "Network bipartivity". In: *Phys. Rev. E* 68 (5 Nov. 2003), p. 056107. DOI: 10.1103/PhysRevE.68.056107. URL: https://link.aps.org/doi/10.1103/PhysRevE.68.056107.

[102] Ernesto Estrada and Juan A Rodriguez-Velazquez. "Spectral measures of bipartivity in complex networks". In: *Physical Review E* 72.4 (2005), p. 046105.

[103] Yurii Vasil'evich Prokhorov. "W. Feller, An Introduction to Probability Theory and Its Applications". In: *Teoriya Veroyatnostei i ee Primeneniya* 10.1 (1965), pp. 204–206.

[104] Rossana Mastrandrea et al. "Enhanced reconstruction of weighted networks from strengths and degrees". In: *New Journal of Physics* 16.4 (2014), p. 043022.

[105] Diego Garlaschelli. "The weighted random graph model". In: *New Journal of Physics* 11.7 (2009), p. 073005.

[106] Andrea Lancichinetti and Santo Fortunato. "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities". In: *Physical Review E* 80.1 (2009), p. 016118.

[107] *Star Wars characters network*. URL: https://github.com/evelinag/StarWars-social-network.

[108] Jérôme Kunegis. *The KONECT Project*. URL: http://konect.cc/.

[109] *'SurpriseMeMore' Github page*. URL: https://github.com/EmilianoMarchese/SurpriseMeMore.

[110] *Infomap module for Python*. URL: https://github.com/mapequation/infomap.

[111] *Louvain module for Python*. URL: https://python-louvain.readthedocs.io/en/latest/.

[112] *Clauset's algorithm for Python*. URL: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.community.modularity_max.greedy_modularity_communities.html#networkx.algorithms.community.modularity_max.greedy_modularity_communities.

[113] *'SurpriseMe' module*. URL: `https://github.com/raldecoa/SurpriseMe`.

[114] *PACO module*. URL: `https://github.com/CarloNicolini/paco`.

[115] *'SurpriseMeMore' module*. URL: `https://pypi.org/project/surprisememore/`.

[116] Jian-Hong Lin et al. "The weighted Bitcoin Lightning Network". In: *arXiv preprint arXiv:2111.13494* (2020).

[117] Seungjin Lee and Hyoungshick Kim. "On the robustness of lightning network in bitcoin". In: *Pervasive and Mobile Computing* 61 (2020), p. 101108.

[118] Stefano Martinazzi and Andrea Flori. "The evolving topology of the Lightning Network: Centralization, efficiency, robustness, synchronization, and anonymity". In: *Plos one* 15.1 (2020), e0225966.

[119] István András Seres et al. "Topological analysis of bitcoin's lightning network". In: *Mathematical Research for Blockchain Economy*. Springer, 2020, pp. 1–12.

[120] Yuwei Guo, Jinfeng Tong, and Chen Feng. "A measurement study of bitcoin lightning network". In: *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE. 2019, pp. 202–211.

[121] Elias Rohrer, Julian Malliaris, and Florian Tschorsch. "Discharged payment channels: Quantifying the lightning network's resilience to topology-based attacks". In: *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE. 2019, pp. 347–356.

[122] Ayelet Mizrahi and Aviv Zohar. "Congestion attacks in payment channel networks". In: *International Conference on Financial Cryptography and Data Security*. Springer. 2021, pp. 170–188.

[123] Marco Conoscenti, Antonio Vetrò, and Juan Carlos De Martin. "Hubs, rebalancing and service providers in the lightning network". In: *Ieee Access* 7 (2019), pp. 132828–132840.

[124] Mark EJ Newman. "The structure and function of complex networks". In: *SIAM review* 45.2 (2003), pp. 167–256.

[125] Mark Ed Newman, Albert-László Ed Barabási, and Duncan J Watts. *The structure and dynamics of networks.* Princeton university press, 2006.

[126] Balaji S. Srinivasan and Leland Lee. *Quantifying Decentralization.* 2017. URL: `https://news.earn.com/quantifying-decentralizatic` `e39db233c28e` (visited on 11/11/2021).

[127] Phillip Bonacich. "Power and centrality: A family of measures". In: *American journal of sociology* 92.5 (1987), pp. 1170–1182.

[128] Stephen P Borgatti. "Centrality and network flow". In: *Social networks* 27.1 (2005), pp. 55–71.

[129] Francisco Aparecido Rodrigues. "Network centrality: an introduction". In: *A mathematical modeling approach from nonlinear dynamics to complex systems*. Springer, 2019, pp. 177–196.

[130] Mark EJ Newman. "A measure of betweenness centrality based on random walks". In: *Social networks* 27.1 (2005), pp. 39–54.

[131] René Pfitzner et al. "Betweenness preference: Quantifying correlations in the topological dynamics of temporal networks". In: *Physical review letters* 110.19 (2013), p. 198701.

[132] Phillip Bonacich. "Some unique properties of eigenvector centrality". In: *Social networks* 29.4 (2007), pp. 555–564.

[133] Jian-Guo Liu et al. "Locating influential nodes via dynamics-sensitive centrality". In: *Scientific reports* 6.1 (2016), pp. 1–8.

[134] James Morgan. "The anatomy of income distribution". In: *The review of economics and statistics* (1962), pp. 270–283.

[135] Paolo Crucitti, Vito Latora, and Sergio Porta. "Centrality measures in spatial networks of urban streets". In: *Physical Review E* 73.3 (2006), p. 036125.

[136] Duncan J Watts and Steven H Strogatz. "Collective dynamics of 'small-world'networks". In: *nature* 393.6684 (1998), pp. 440–442.

[137] Vito Latora and Massimo Marchiori. "Efficient behavior of small-world networks". In: *Physical review letters* 87.19 (2001), p. 198701.

[138] LAN Amara et al. "Classes of small-world networks". In: *The Structure and Dynamics of Networks*. Princeton University Press, 2011, pp. 207–210.

[139] Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. "Power-law distributions in empirical data". In: *SIAM review* 51.4 (2009), pp. 661–703.