UNIVERSITY OF FRIBOURG

MASTER THESIS

Giant connected component in networks

Author:
Benoît RICHARD

Supervisors:
Dr. Guiyuan SHI
Prof. Yi-Cheng ZHANG

Theoretical Interdisciplinary Physics Group
Department of Physics

September 27, 2018

University of Fribourg

Abstract

Faculty of Science
Department of Physics

Master Project

Giant connected component in networks

by Benoît RICHARD

Write the abstract

Contents

List of Symbols

Single layer networks

Symbol	Description	Math. definition
c	Expectation value for the degree	$\mathbb{E}[\deg v]$
$\deg v$	Degree of vertex v	
$\mathbb{E}[\dots]$	Expectation value	
$g_0(z)$	Generating function for the degree of uniformly chosen nodes	$\sum_{k=0}^{\infty} p_k z^k$
$g_1(z)$	Generating function for the degree of nodes reached by following an edge	$\sum_{k=0}^{\infty} q_k z^k$
k_i	Degree of vertex i	
m	Number of edges in the network	
n	Number of nodes in the network	
$N(v)$	Neighborhood of a vertex v	
p_{ij}	Probability that vertices i and j are connected	
p_k	Probability that a random node has degree k	$P_0(\deg v = k)$
$P_0(\dots)$	Probability starting from a uniformly chosen node	
$P_1(\dots)$	Probability starting from a node reached by following an edge	
q_k	Probability that a node reached by following a edge has degree $k+1$	$P_1(\deg v = k+1)$
r_k	Probability that a random node of the GCC has degree k	$P_0(\deg v = k \vert v \in \mathrm{GCC})$
S	Fraction of the network which is part of the GCC in the large n limit	$P_0(v \in \mathrm{GCC})$
u	Probability that a node reached by following an edge is not part of the GCC	$P_1(v \notin \mathrm{GCC})$
v	Random variable representing a vertex chosen in a network, either uniformly or by following an edge depending of the context	
α	Exponent of a power law distribution	
$\zeta(\alpha)$	Riemann zeta function	

Multiplex networks

For multiplex networks, the convention we follows to indicate a value refer to layer i is to add an upper index (i) if the quantity has a lower index and to add a lower index i otherwise.

Symbol	Description	Math. definition
C	Critical region in the parameters space	
L	Number of layers of the multiplex network	
N	Number of parameters determining the degree distributions of a multiplex network	
$N_i(v)$	Neighborhood of a vertex v in layer i	
$g_0^{(i)}(z)$	Generating function for the degree of uniformly chosen nodes in layer i	$\sum_{k=0}^{\infty} p_k^{(i)} u_i^k$
$g_1^{(i)}(z)$	Generating function for the degree of nodes reached by following an edge in layer i	$\sum_{k=0}^{\infty} q_k^{(i)} u_i^k$
$p_k^{(i)}$	Probability that a uniformly chosen node has degree k in layer i	$P_0^{(i)}(\deg v = k)$
$P_0^{(i)}(\dots)$	Probability for an event in layer i starting from a uniformly chosen node	
$P_1^{(i)}(\dots)$	Probability for an event in layer i starting from a node reached by following an edge in layer i	
$q_k^{(i)}$	Probability that a node reached by following a edge has degree $k+1$	$P_1^{(i)}(\deg v = k+1)$
u_i	Probability that a node reached by following an edge in layer i is not part of the GVC	$P_1^{(i)}(v \notin GVC)$
$\mathbf{u}$	Vector of all u_i	$(u_1, u_2, \dots, u_L)$
$\mathbf{u}_T$	Trivial solution for $\mathbf{u}$	$(1, 1, \dots, 1)$
λ_j	One of the N parameters determining the degree distributions of a multiplex network	
$\boldsymbol{\lambda}$	Vector of all λ_j	$(\lambda_1, \lambda_2, \dots, \lambda_N)$

Todo list

Chapter 1

Single layer networks

layer networks⟩?

1.1 Introduction

Many systems in real world can be conceptually represented as objects being connected to others. Such representation is called a *network*. For example, a power grid can be represented as stations connected by power lines, as shown in fig. 1.1 for the power grid of western USA. The concept of network does not require the object or the links between them to be physical. As an example, we can represent collaboration as a network: two people are connected if they did collaborate. Such network is shown in fig. 1.2 for collaboration of jazz musicians. This implies that a very different systems can be represented as networks, from cities connected by road (see fig. 1.3 for E-road network) to protein connected if they interact (see fig. 1.4 for a network of human proteins). Insight on fundamental properties of networks may shed light on a very broad range of problems. To gain such insights, theoretical studies of general networks, such as the one presented in this thesis, is required.

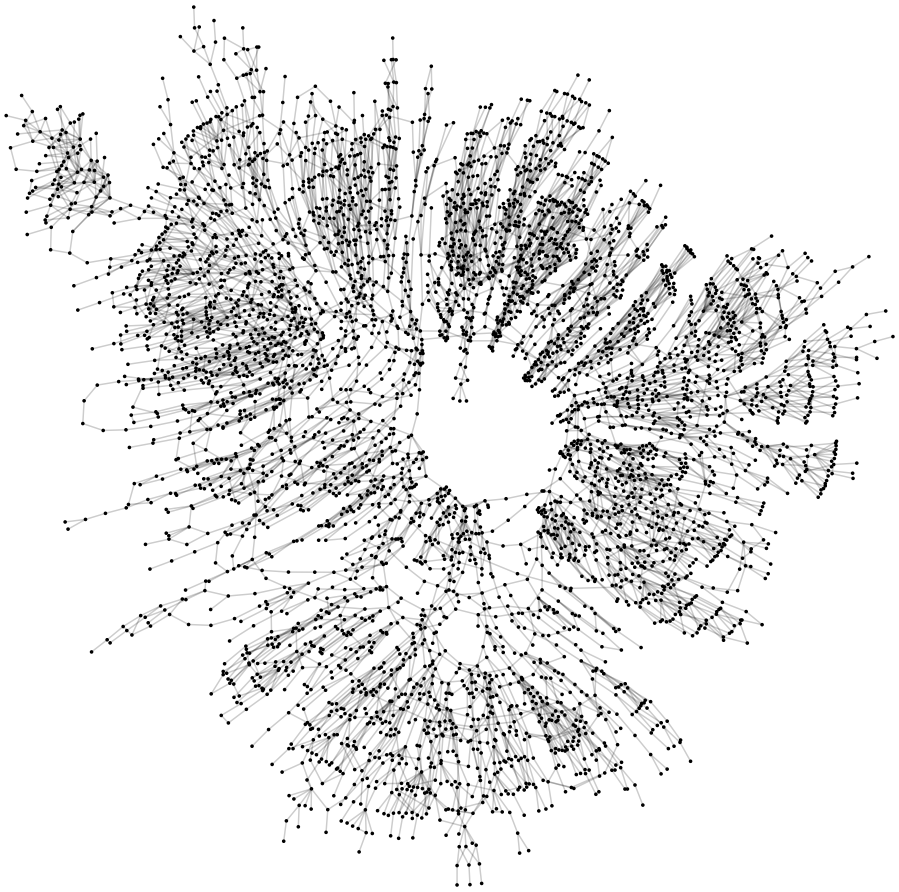
Mathematically, networks are represented as *graphs*. A graph is an object composed of a set V of *vertices* (also referred to as nodes) and a set of *edges* E. An edge is characterized by the fact that it connects two vertices together, which in mathematical terms translate to the fact that an edge can be written as a pair of vertices or equivalently $E \subset \{(v_1, v_2) | v_1, v_2 \in V\}$. Many extensions of this model exist, for example edges may have a direction (*directed graph*), implying that $(v_1, v_2) \neq (v_2, v_1)$, or edges can carry a value (*weighted graph*).

In this first chapter we focus on the standard unweighted and undirected networks, even restricting our theoretical analysis to the case of infinitely large networks as it allows for several convenient simplifications while still capturing important feature networks. In Chapter 2 we study an extension of this network category, multiplex networks, which consists in allowing distinct type of edges between the vertices.

1.2 Configuration model

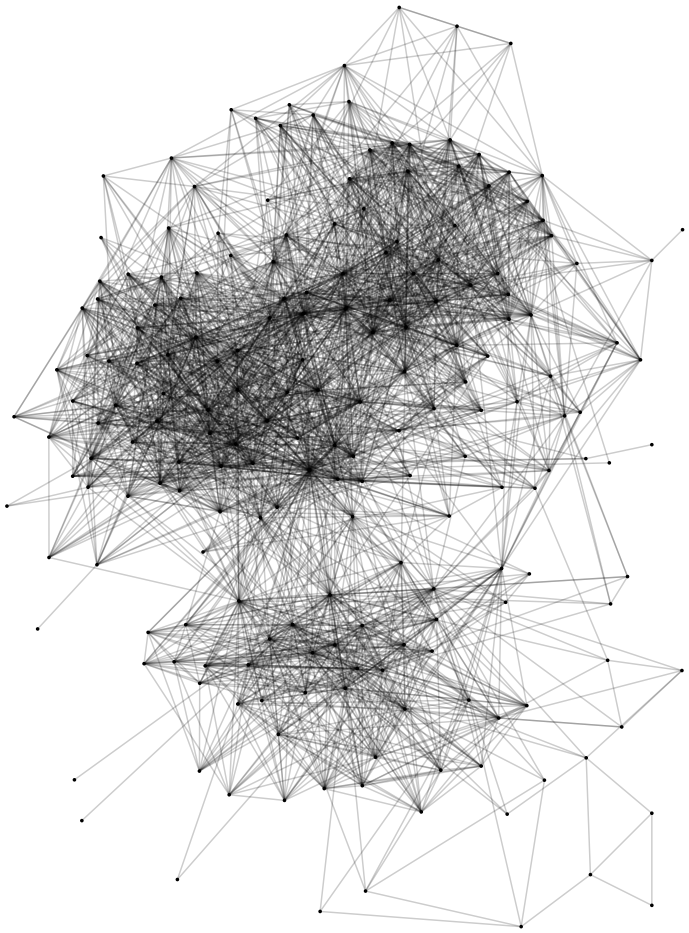
iguration model⟩ Since we are interested in fundamental properties of networks, we need to abstract from the specificity of one network. To do so, we consider that networks are fully determined by their *degree distribution* $\{p_k\}$ where p_k is the probability for a node chosen randomly and uniformly to have degree k. Since knowing how a network can be constructed is useful both conceptually and to perform computation on properties of the network, we now present an algorithm called *configuration model* [1] that sample uniformly the space of all network with a given degree distribution.

Consider a network with n vertices with a given degree distribution $\{p_k\}$. If we cut every edges in two, every vertex keep a number of *stubs* (half-edges) equal to



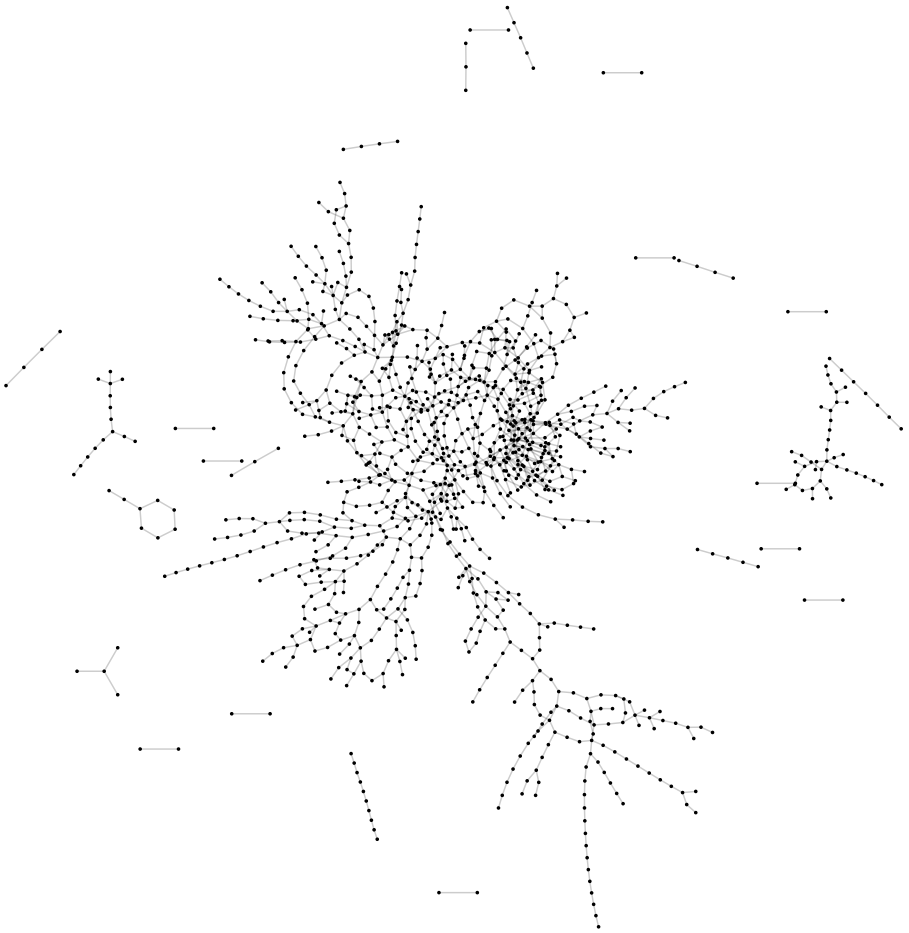
estern US powergrid⟩

FIGURE 1.1: Power grid network of the Western States of USA. Nodes represent electrical stations (generator, transformator, substation) and edges represent power supply lines. Data retrieved from the Konect database [4] (Konect code UG).



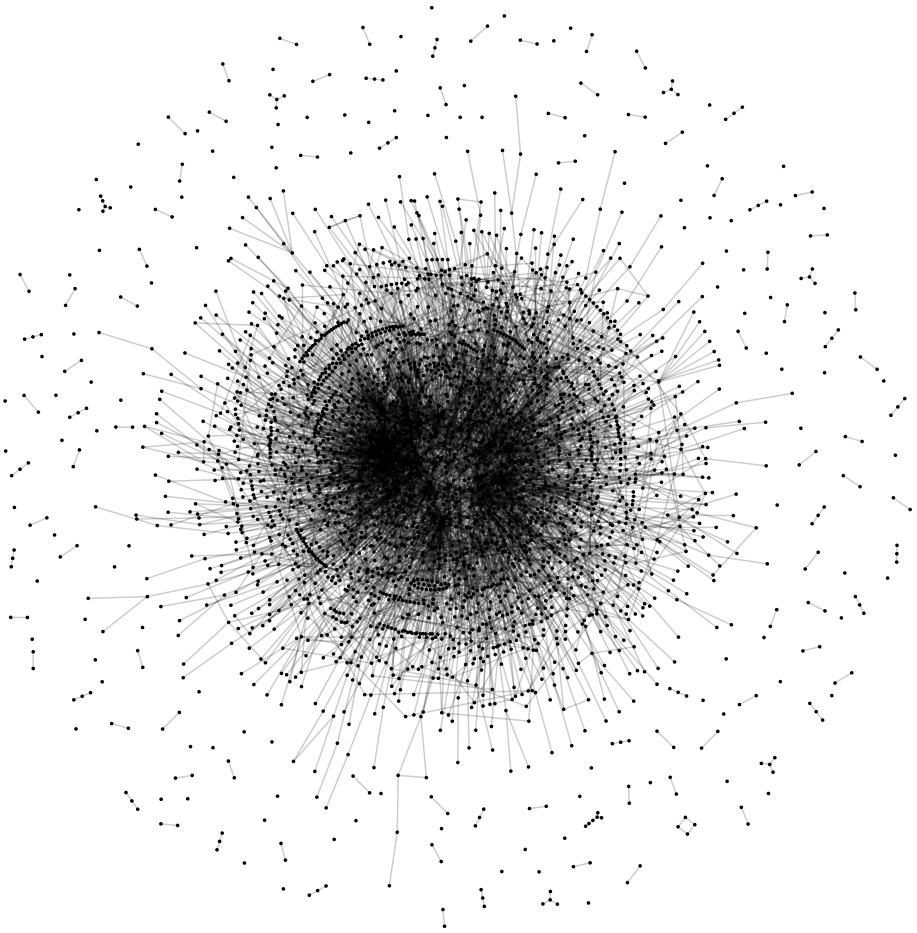
collaborations⟩

FIGURE 1.2: Collaboration network of jazz musicians as of 2003. Nodes represent musicians and edges represent the fact that the two musicians had played in the same band. Data retrieved from the Konect database [4] (Konect code JZ).



e: Network euroroad⟩

FIGURE 1.3: International E-road network, a network of road situated mainly in Europe. Nodes represent cities and edges represent E-roads connecting them. Data retrieved from the Konect database [4] (Konect code ET).



human proteins⟩

FIGURE 1.4: Network of protein-protein interaction in humans. Nodes represnt proteins and edges a binary interaction. Data retrieved from the Konect database [4] (Konect code MV).

(A) Nodes and theirs stubs (B) Stubs are randomly connected

onfiguration model⟩?

FIGURE 1.5: Schematic representation of the configuration model.

its degree. The resulting set of vertices and stubs is independent of the network structure, but common for all networks with the same degree distribution. The idea of this algorithm is thus to start from this state, a set of nodes with stub degree distribution $\{p_k\}$. Then each stub is bond to another chosen uniformly amongst other stubs to form all edges of the network. By construction, the produced network has degree distribution $\{p_k\}$.

The fact that stubs are paired uniformly and independently is important in that it implies that the vertex reached by following a random edge does not dependant, in probability, on the vertex at the other end of the edge.

Think about that and the need for that for GCC calculation

1.3 Self-edges and multi-edges

Using the insight given by the algorithm, we can compute the probability that a node is connected to itself, thus making a so-called *self-edge*. In a network with m edges, there are $2m$ stubs. The probability p_{ii} that a stub of vertex i connect to another stub of the same vertex is thus

$$p_{ii} = m \frac{\binom{k_i}{2}}{\binom{2m}{2}} = \frac{k_i(k_i-1)}{2(2m-1)}, \tag{1.1}$$ {?}

where k_i is the degree of vertex i. In the limit of large n, the number of vertices with degree k is equal to np_k and as a consequence the total number m of edges is equal to

$$m = \frac{1}{2}\sum_{k=0}^{\infty} nkp_k = \frac{n}{2}\mathbb{E}[\deg v]\,, \tag{1.2}$$ {?}

with $\mathbb{E}[\ldots]$ denoting the expectation value. The average number of self-edges is therefore asymptotically constant as n becomes large, so the fraction of vertices having self edges goes to zeros as n grows and we can safely consider the generated network has no self-edges at all.

Similarly, we find that the probability p_{ij} that two vertices i and j are connected is equal to

$$p_{ij} = m \frac{k_i k_j}{\binom{2m-2}{2}} = \frac{k_i k_j}{2m-1}. \tag{1.3}$$ {?}

The probability to have two or more edges between the vertices i and j is equal to the probability that i and j are connected and that they remain so after we remove one edge between them. The probability for them to be connected with one edge less is the same as p_{ij} but with one edge less in total and one stub less at both i and j, giving

$$\frac{(k_i-1)(k_j-1)}{2m-3}. \tag{1.4}$$ {?}

In consequence we find the probability to have at least two edges between i and j to be

$$p_{ij}\frac{(k_i-1)(k_j-1)}{2m-3}, \tag{1.5}$$ {?}

giving the average number of multi-edges

$$\frac{\sum_{ij} k_i k_j (k_j-1)(k_i-1)}{2(2m-1)(2m-3)} \approx \frac{1}{8m^2}\sum_i k_i(k_i-1)\sum_j k_j(k_j-1) \tag{1.6}$$ {?}

$$= \frac{1}{2}\left[\frac{\mathbb{E}\left[(\deg v)^2\right]-\mathbb{E}[\deg v]}{\mathbb{E}[\deg v]}\right]^2. \tag{1.7}$$ {?}

The approximation arise as in the limit of large n we have $2m-3 \approx 2m-3 \approx 2m$ as m scale proportionally to n. As in the case of self-edges, the number of multi-edges is asymptotically constant and we can therefore consider that the generated networks has no multi-edges.

Since $\{p_k\}$ is a probability distribution, it is independent of the number of nodes n of the network. Therefore for any degree distribution, we can consider the limit for large n, which we do as it allows several mathematical simplifications of the problem as outlined below. For sufficiently large networks, the difference between the large n limit and the actual network is small and can thus safely be neglected.

A network having this two properties, absence of self-edges and of multi-edges, is said to be a *simple graph*. Since for n large enough all networks in the context of the configuration model have approximately these two properties, we always consider that the networks are simple graphs in the remaining of this thesis.

1.4 Uniformity of network space sampling

In the previous section we demonstrated that the configuration model asymptotically produces simple graphs in the limit of large n. This is necessary to prove the claims we make in Section 1.2 that the configuration uniformly sample the space of networks with a given degree distribution. To prove that claim is equivalent to prove that each different network appears with the same probability.

The following calculation is not correct, nodes with the same degree are indistinguishable as well. Correct and note the disagreement with Newman

A *matching* of the stubs is a possible way to bond $2m$ distinguishable stubs. By construction the pair of stubs that are bound are chosen uniformly and thus each matching appears with the same probability than any other. However, in practice the stubs of a single vertex are not distinguishable and several matching leads to the same network. Therefore there is $k!$ way of arranging the stubs of a vertex of degree

k, resulting in a total of

$$\prod_{i=1}^{n} k_i! \tag{1.8}$$ {?}

matchings corresponding to the same network, where k_i is the degree of vertex i. This quantity only depends on the degree distribution, thus all networks with the same degree distribution have the same number of different matchings. Since each matching appears with the same probability, we can conclude that each network with a given degree distribution appears with the same probability in the configuration model.

In a sense the configuration model is therefore optimal for the point of view we adopt in this thesis. Indeed we consider a network fully determined by its degree distribution and all network with a common degree distribution are equal with regard to the configuration model as it sample them uniformly. We could in fact present the configuration model as a consequence of the requirement that we want a sampling method introducing no unnecessary constrain on the produced networks. See Section 15.2 of [1] or [?] for discussions of that point of view.

sing ref: maximum entropy paper

1.5 Excess degree distribution

As we will see below, while we consider that a network is fully determined by its degree distribution, considering vertices reached by following an edge gives valuable insights on the network structure. We call such vertex a *first neighbor* vertex and we denote $P_1(\dots)$ the probability associated with a first neighbor, while we denote $P_0(\dots)$ the probability associated with uniformly chosen vertices[1]. We can define the *excess degree distribution* $\{q_k\}$ as

$$q_k = P_1(\deg v = k+1), \qquad \forall k \in \mathbb{N}. \tag{1.9}$$ {?}

The probability q_k correspond to a first neighbor having degree $k+1$, or equivalently to the probability to have k edges other than the on used to reach the node in the first place, hence the name excess degree distribution.

The excess degree distribution can be computed explicitly by noting that a stub has the same probability to be connected to any if the other $2m-1$ stubs, thus the probability that this stub is connected to a given node of degree k is $k/(2m-1)$. Multiplying by the total number of node of degree k, np_k in the large n limit,gives the probability that a given node is attached to a node of degree k as

$$\frac{k}{2m-1} np_k = \frac{kp_k}{\mathbb{E}[\deg v]}. \tag{1.10}$$ {?}

Now q_k is the probability that a first neighbor has degree $k+1$, so we can conclude

$$q_k = \frac{(k+1)p_{k+1}}{\mathbb{E}[\deg v]}. \tag{1.11}$$

qk as function

[1] In principle $P_j(\dots)$ could be defined, corresponding to the probability associated with vertices reached after following j edges.

1.6 Generating functions

ting functions)? A powerful way of representing a degree distribution (or any discrete probability law) is the *generating function* of the distribution. For a degree distribution $\{p_k\}$ it is defined as the function

$$g_0(z) = \sum_{k=0}^{\infty} p_k z^k. \tag{1.12}$$ Definition of g0

Observe that the derivative with respect to z of $g_0(z)$ is

$$g_0'(z) = \sum_{k=0}^{\infty} p_k k z^{k-1}. \tag{1.13}$$ Derivative of g0

Comparing with definition of the expectation value we find

$$\mathbb{E}[\deg v] = \sum_{k=0}^{\infty} p_k k = g_0'(1). \tag{1.14}$$ Expectation value a

Therefore, the generating function is sufficient to know the expectation value of a distribution. This result can be generalized: if we look at the second derivative of the generating function, by differentiating eq. (1.13), we find

$$g_0''(1) = \sum_{k=0}^{\infty} p_k k(k-1) = \sum_{k=0}^{\infty} p_k k^2 - \sum_{k=0}^{\infty} p_k k \tag{1.15}$$ {?}

$$= \mathbb{E}\left[(\deg v)^2\right] - \mathbb{E}[\deg v]\,. \tag{1.16}$$ {?}

Thus we can write

$$\mathbb{E}\left[(\deg v)^2\right] = g_0''(1) + g_0'(1) = \frac{\partial}{\partial z}\left(z g_0'(z)\right)\Big|_{z=1}, \tag{1.17}$$ Second moment as de

which means that the second moment is fully determined by the generating as well. In fact such formula exists for all moments, but we do not present it here as we only need the first two moments. The second moment is used to compute the variance of the degree distribution as follow

$$\mathrm{var}[\deg v] = \mathbb{E}\left[(\deg v)^2\right] - \left(\mathbb{E}[\deg v]\right)^2 \tag{1.18}$$ {?}

$$= \frac{\partial}{\partial z}\left(z g_0'(z)\right)\Big|_{z=1} - \left(g_0'(1)\right)^2. \tag{1.19}$$?Variance as derivat

All these properties do not depend on the degree distribution p_k to which the generating function is associated and apply to any discrete probability distribution. One other important distribution for network is the excess degree distribution $\{q_k\}$, we define its generating function g_1 as

$$g_1(z) = \sum_{k=0}^{\infty} q_k z^k. \tag{1.20}$$ Definition of g1

Inserting eq. (1.11) in this definition, we get

$$g_1(z) = \frac{1}{\mathbb{E}[\deg v]} \sum_{k=0}^{\infty} (k+1) p_{k+1} z^k = \frac{g_0'(z)}{g_0'(1)}, \tag{1.21}$$

g1 as a functio

where we used eq. (1.32) and eq, (1.14) for the last equality.

1.7 Erdos-Renyi networks

In this thesis we focus on three types of theoretical networks, beside some real examples: Erdos-Renyi networks, scale-free networks and geometric networks.

An Erdos-Renyi networks is characterized by the fact that it can be grown as follow: for each pair of nodes i and j, add an edge with probability p. To find the degree distribution in such network, first notice that the expected degree, usually denoted c for Erdos-Renyi network, is equal to the number of other vertices multiplied by the probability to be connected to each of them, i.e.

$$c = \mathbb{E}[\deg v] = (n-1)p. \tag{1.22}$$

{?}

We generally use c as the parameter defining an Erdos-Renyi network, rather than p, since it makes more to keep c constant when n becomes large, rather than p.

The probability for a node to have degree k is

$$p_k = \binom{n-1}{k} p^k (1-p)^{n-1-k}, \qquad \forall k \in \mathbb{N}. \tag{1.23}$$

?Poisson degree

We recognize a binomial degree distribution for $n-1$ trials with success probability p. In the limit of large n we can approximate such distribution by a Poisson distribution with parameter $c = (n-1)p$

$$p_k \approx \frac{c^k}{k!} e^{-c}. \tag{1.24}$$

pk for Erdos-Re

The parameter c is the expected degree in the network, it is proportional to $n-1$ rather than n because we only tries to bind each vertex with each other, and not with itself, making a total of $n-1$ trials.

Inserting the degree distribution in the definition of the generating function (1.12), we recognize Taylor series representing the exponential function and thus we get

$$g_0(z) = e^{-c} \sum_{k=0}^{\infty} \frac{z^k c^k}{k!} = e^{-c} e^{cz} = e^{c(z-1)}. \tag{1.25}$$

?g0 for ER netwo

Taking the derivative and inserting in eq. (1.20) yields the generating function for the excess degree distribution

$$g_1(z) = e^{c(z-1)}, \tag{1.26}$$

{?}

which appears to be equal to $g_0(z)$.

1.8 Geometric networks

Geometric networks have a geometric degree distribution

Ask Guiyuan which version to use and if there is a some motivation in using this geom. dist.

1.9 Scale-free networks

The degree distribution of a so-called scale-free network follows a power law with exponent α

$$p_k = \frac{k^{-\alpha}}{\zeta(\alpha)}, \qquad \forall k \in \mathbb{N}^*, \tag{1.27}$$

?Power law degree di

where $\zeta(\alpha)$ is the Riemann zeta function and $p_0 = 0$. This kind of networks is interesting as many real networks exhibits power law tail in their degree distribution . However, power law distribution are mathematically more complicated than the two previous examples as their generating function can not be represented in term of elementary function. The best we can do is introducing the *polylogarithm* $\mathrm{Li}_\alpha(z)$

Add examples

$$\mathrm{Li}_\alpha(z) = \sum_{k=1}^{\infty} k^{-\alpha} z^k. \tag{1.28}$$

Definition of polyl

The polylogarithm is a generalization of the Riemann zeta function, as can be seen by the fact that for $z = 1$ we have

$$\mathrm{Li}_\alpha(1) = \sum_{k=1}^{\infty} k^{-\alpha} = \zeta(\alpha). \tag{1.29}$$

Polylogarithm of 1

Credit the guy the polylog code comes from. Maybe say a bit more about polylog/reimplement polylog using intergation

With that notation, the generating function $g_0(z)$ for scale-free networs can be written

$$g_0(z) = \sum_{k=1}^{\infty} \frac{k^{-\alpha}}{\zeta(\alpha)} z^k = \frac{\mathrm{Li}_\alpha(z)}{\zeta(\alpha)}. \tag{1.30}$$

?Generating function

While no simple expression exist for the polylogarithm, its formal definition (1.28) is sufficient to compute its derivative

$$\frac{\partial}{\partial z}\mathrm{Li}_\alpha(z) = \sum_{k=1}^{\infty} k^{-\alpha+1} z^{k-1} = \frac{1}{z}\mathrm{Li}_{\alpha-1}(z)\,. \tag{1.31}$$

?Derivative of the p

We therefore find

$$g_0'(z) = \frac{\mathrm{Li}_{\alpha-1}(z)}{z\zeta(\alpha)} \tag{1.32}$$

Derivative of g0 fo

that we can insert in eq. (1.21) to get

$$g_1(z) = \frac{\mathrm{Li}_{\alpha-1}(z)}{z\zeta(\alpha-1)}. \tag{1.33}$$

{?}

We used eq. (1.29) to perform the simplification $g_0'(1) = \zeta(\alpha-1)/\zeta(\alpha)$. Thanks to eq. (1.14), we know that $g_0'(1) = \mathbb{E}[\deg v]$. Thus we have

$$\mathbb{E}[\deg v] = \frac{\zeta(\alpha-1)}{\zeta(\alpha)}. \tag{1.34}$$

Mean degree in

Since we are not considering the analytic continuation of the zeta function, but only its real sum description given in eq. (1.29), $\zeta(\alpha)$ diverges for $\alpha < 1$. We can therefore distinguish three cases. First for $\alpha \leq 2$, eq. (1.34) diverges and the mean degree always goes to infinity as the network grows. The name scale free comes from that fact, as there is no *typical scale* for the degrees and greater degrees can always happen with probability too high to be ignored.

In the case $2 < \alpha \leq 3$, the mean degree is finite, but its second moment is not. It can be seen by calculating explicitly using using eq. (1.17),

$$\mathbb{E}\big[(\deg v)^2\big] = \frac{\partial}{\partial z}\left(z g_0'(z)\right)\Big|_{z=1} = \frac{\partial}{\partial z}\left(\frac{\mathrm{Li}_{\alpha-1}(z)}{\zeta(\alpha)}\right)\Big|_{z=1} \tag{1.35}$$

{?}

$$= \frac{\mathrm{Li}_{\alpha-2}(1)}{\zeta(\alpha)} = \frac{\zeta(\alpha-2)}{\zeta(\alpha)}. \tag{1.36}$$

{?}

The divergence of the second moment come from the fact that the factor $\zeta(\alpha-2)$ diverges for $\alpha \leq 3$. In consequence, the variance of the degree distribution diverges, which means the degree are very widely distributed.

A nicer interpretation arises from a consequence of eq. (1.21), namely

$$g_1'(1) = \frac{g_0''(z)}{g_0'(z)}. \tag{1.37}$$

{?}

A similar calculation as for the second moment shows that for $\alpha \leq 3$, this diverges too. Hence, for $2 < \alpha \leq 3$ the degree distribution is not strictly speaking scale free, but the excess degree distribution is.

Finally for $\alpha > 3$, the two first moment are finite and only higher moments diverge, which does not implies any peculiar behavior.

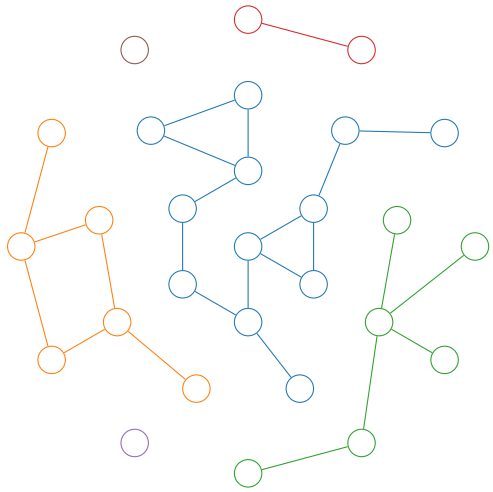
1.10 Giant connected component

connected component⟩?

1.10.1 Size of the GCC

An interesting property of a network is the presence and size of *connected component*. A set of nodes is said to be connected if there is a path formed of successive edges from any of its node to any other. All networks can be divided in connected components such that all nodes are element of exactly one component, as is exemplified in fig. 1.6. The connectedness of network is crucial in many real world realisations of networks. In particular any logistic network, such as power grid networks, rail road network or the internet network, is functional only if it is able to transfer goods or services (electricity, passengers or informations) from any node to any other.

Insight on the property of networks can be found by studying the case where the fraction of the network forming its biggest component does not vanish in the large n limit. This component is called the *giant connected component* (GCC). The first question to answer about it is: what will be the size of the giant connected component ?



cted components⟩

FIGURE 1.6: Scheme of a graph with six connected components, each drawn with a different color.

To determine this consider a connected component C of the network. We define u the probability that a node reached by following an edge is not part of C. We can therefore write the probability S that a randomly chosen vertex is part of C as

$$S = 1 - P_0(w \notin C \; \forall w \in N(v)) \tag{1.38}$$ {?}

$$= 1 - \sum_{k=0}^{\infty} P_0(w \notin C \; \forall w \in N(v) | \deg v = k) P_0(\deg v = k). \tag{1.39}$$ {?}

The probability $P_0(w \notin C \; \forall w \in N(v) | \deg v = k)$ is the probability that no neighbors of a node with degree k are part of the component C. This in turn is the probability that by following k independent edges[2] we find each time a node which is not part of C. This observation allow us to write

$$P_0(w \notin C \; \forall w \in N(v) | \deg v = k) = [P_1(w \notin C)]^k = u^k. \tag{1.40}$$ Probability that a

In the last equality we introduce the probability u for a first neighbor not to be part of C,

$$u = P_1(w \notin C). \tag{1.41}$$?Definition of u?

Noting that $P_0(\deg v = k) = p_k$, we find finally

$$S = 1 - \sum_{k=0}^{\infty} u^k p_k \tag{1.42}$$ {?}

$$= 1 - g_0(u). \tag{1.43}$$ {?}

We now have a compact expression for S in terms of u and the generating function of the degree distribution g_0. To determine u we observe that if a vertex is not

[2] Edges are independant by construction of the configuration model.

part of C, none of its neighbors is either. Thus we can write

$$u = P_1(w \notin C \; \forall w \in N(v)) \tag{1.44}$$ {?}

$$= \sum_{k=0}^{\infty} P_1(w \notin C \; \forall w \in N(v) | \deg v = k) P_1(\deg v = k) \tag{1.45}$$ {?}

$$= \sum_{k=0}^{\infty} u^k q_k \tag{1.46}$$ {?}

$$= g_1(u), \tag{1.47}$$ {?}

where we use eq. (1.40) again together with the fact that by definition of the excess degree distribution $q_k = P_1(\deg v = k)$.

We end up with two equations to describe the GCC size

$$S = 1 - g_0(u) \tag{1.48}$$ Single layer S

$$u = g_1(u). \tag{1.49}$$ Single layer u

If we can solve the second one we immediately get the size of the connected component C. However eq. (1.49) only gives u implicitly and its form strongly depends on the degree distribution, therefore no general analytical solutions can be given. From its graphical representation, shown in fig. 1.7, we can see that it has at most two solutions. First the trivial solution $u = 1$ is always present, as by the definition (1.20) of $g_1(z)$, we have $g_1(1) = 1$, implying $S = 0$. The components described by this regime are not giant connected components, as their size S vanish in the large n limit.

Then in some cases, another solution exists with $u < 1$ and $S > 0$. This solution correspond to the GCC. Note that this implies that if a network admit multiple GCC, they must all be of the same relative size S.

Prove that the GCC must be unique

Add something about the fact we will refer to u and S thinking about the GCC

To see when a GCC exists, observe that, as can be seen on fig. 1.7, the $g_1(z)$ curve must "go below" the identity curve at $z = 1$ to create a non trivial solution. This requirement means that the slope of $g_1(z)$ at $z = 1$ must be greater than the slope of the identity, which is 1. In term of the derivative of $g_1(z)$ the condition is thus

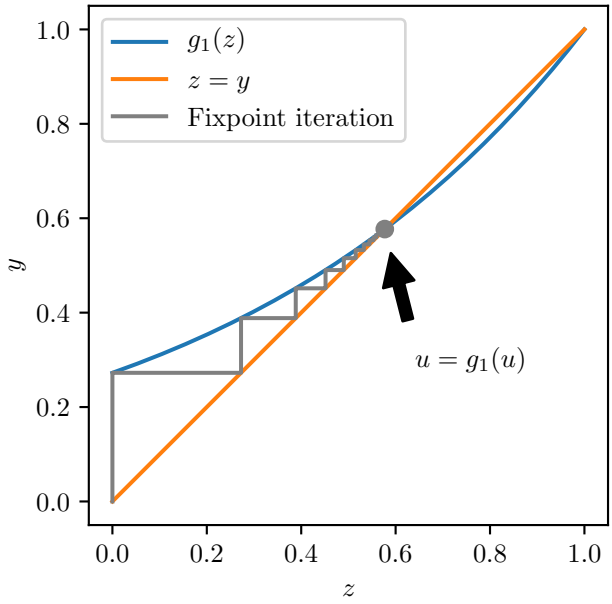
$$g_1'(z) > 1 \tag{1.50}$$ Boundary condit

to have a non trivial solution to eq. (1.49).

Finally eq. (1.49) can be solved numerically by noticing that the solution u is a fixpoint of the function $g_1(z)$. Since all coefficient in eq. (1.20) are non negative, $g_1(z)$ and all its derivative are positive for $z > 0$. As a consequence starting from $z_0 = 0$, the sequence z_k defined by the iteration

$$z_{k+1} = g_1(z_k) \tag{1.51}$$ Single layer fi

always converges toward the smallest solution of eq. (1.49).



(u) graphically⟩

FIGURE 1.7: Graphical representation of eq. (1.49) for an Erdos-Renyi network with $c = 1.3$. Gray solid line represent the successive steps of the fixpoint iteration.

1.10.2 Algorithm to find the connected components

Since we are working in the limit of large n, the networks we are generating and analysing have large n as well. Therefore the algorithm we use must be designed with some care to avoid consuming to much computing time, which would make them impractical to use. This motivate us to present the algorithm we use here.

To find all connected components of a network we proceed as follow

1. Add all nodes to the set $\mathcal{S}_{\text{unprocessed}}$ of unprocessed nodes.
2. Remove one node from $\mathcal{S}_{\text{unprocessed}}$ and add it to the set $\mathcal{S}_{\text{queued}}$ of queued nodes.
3. Start a new component C.
4. Remove one node from $\mathcal{S}_{\text{queued}}$ nodes and name it v.
5. Add v to the current component C.
6. Add all unprocessed neighbors of v to $\mathcal{S}_{\text{queued}}$.
7. If $\mathcal{S}_{\text{queued}}$ is not empty go to 4, else store the component C and continue.
8. If $\mathcal{S}_{\text{unprocessed}}$ is not empty go to 2, else terminate.

This algorithm goes through each node exactly once and is thus of complexity $\mathcal{O}(n)$ which is the optimal complexity since all nodes must be associated to a component.

However, to ensure that the algorithm is fast we must be to efficiently find all neighbors of a node. To do that we represent the network as an *adjacency list*: each node is given an index i and the adjacency list A_i contains all the neighbors of i. The whole network is thus represented as a list of adjacency list $A = (A_1, \ldots, A_n)$.[3]

[3]For better performance during the creation of the network, our implementation goes a step further and actually request sorted adjacency lists.

Other representation of networks exist, which are more convenient and efficient for some purposes. However we do not use them in this thesis and we stick to the adjacency list representation.

In fig. 1.8 we compare the fraction of the network occupied by the biggest component found using this algorithm with the numerical solution of eq. (1.48) and (1.49) computed using the fixpoint iteration (1.51). As we can see the agreement between the two is very good for large n.

Tidy the graphs and add scale-free

Add graph with varying n to show the effect

1.10.3 Degree distribution in the GCC

ribution in the GCC)? Per Bayes theorem we have for two random events A and B

$$P(A|B) = P(B|A)\frac{P(A)}{P(B)}. \tag{1.52}$$?Bayes theorem?

We can apply it to compute the probability r_k that a vertex in the GCC has degree k

$$\begin{aligned} r_k &= P\left(\deg v = k | v \in GCC\right) && (1.53)\ \{?\}\\ &= P(v \in GCC|\deg v = k)\frac{P(\deg v = k)}{P(v \in GCC)} && (1.54)\ \{?\}\\ &= \frac{p_k}{S}\left(1 - P(v \notin GCC|\deg v = k)\right) = \frac{p_k}{S}(1-u^k). && (1.55) \end{aligned}$$ Degree distribu

To get the final result we used eq. (1.40). This result was previously presented more generally and following a very different path in [2].

Therefore we see that considering a vertex in the GCC biases the probability that it has degree k by a factor $(1-u^k)/S$ as compared to choosing a vertex uniformly in the network. Since both u and S are smaller than 1, the net effect is to lower the proportion of low degree vertices in the GCC and thus to increase the proportion of high degree vertices. This result can be understood intuitively since in the configuration model the stubs are connected independently. Therefore each edge of a node increases the probability that this node is connected to the GCC, making high degree node over represented in the GCC.

1.11 Generating connected networks

connected networks)?

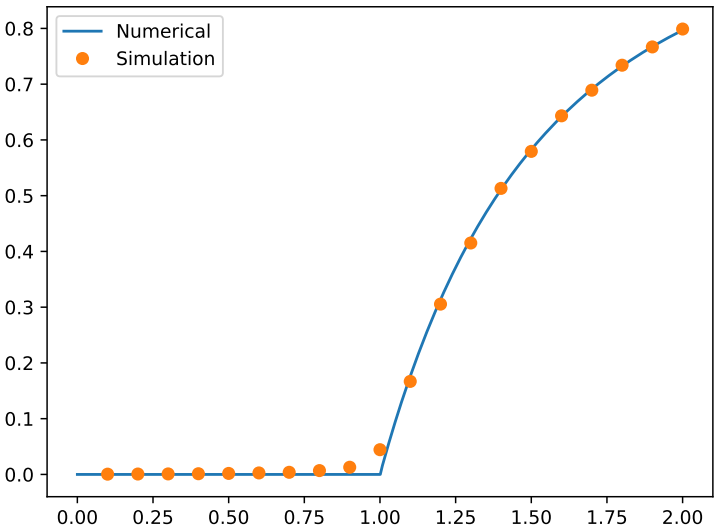
1.11.1 Algorithm

o something with
ith this information

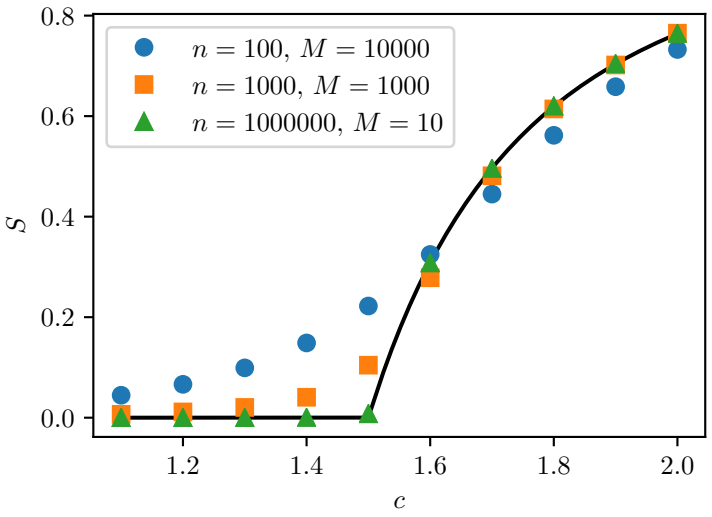
Algorithm previously in [3]

The knowledge of the degree distribution in the GCC can be used generate a connected component of a given degree distribution r_k as follow: we first determine a degree distribution p_k fulfilling eq. (1.55) for some target degree distribution r_k. Then we generate a network with degree distribution p_k using the configuration model. Finally we take its GCC as our connected network. By construction the vertices in the GCC will have degree distribution r_k. Determining the factors p_k is not immediate however since u is an unknown which is itself a function of p_k. We propose an algorithm to determine it numerically.

(A)



(B)



er S simulation⟩

FIGURE 1.8: Numerical solution of eqs. (1.48) and (1.49), together with results on simulated networks. Results of simulations are average over 10 runs and the network size was set to 10^4 nodes. Simulations ran for several seconds in total, indicating that much larger network should be easily usable. (A) Poisson degree distribution. (B) Geometric degree distribution.

First we isolate p_k from eq. (1.55) to get

$$p_k = S\pi_k(u), \quad \text{with} \quad \pi_k(z) = \frac{r_k}{1-z^k} \tag{1.56}$$ {?}

Inserting this in the expression (1.49) for u, we get

$$u = \frac{\sum_{k=1}^{\infty} k\pi_k(u)u^{k-1}}{\sum_{k=1}^{\infty} k\pi_k(u)}. \tag{1.57}$$ Fixpoint equati

Therefore u is a fixpoint of the function

$$\mu(z) = \frac{\sum_{k=1}^{\infty} k\pi_k(z)z^{k-1}}{\sum_{k=1}^{\infty} k\pi_k(z)}, \tag{1.58}$$ Defition of mu

which is fully determined by the GCC degree distribution r_k. Note that for $r_1 = 0$, we have the fixpoint $u = 0$ and $p_k = r_k$ for all k. This is consistent with the fact that small component of a network produced with the configuration model have a probability 0 to have loop [1]. Indeed if $p_1 = 0$ all components must have loops, therefore the probability to have small components is 0 as well.

On the other hand $r_1 > 0$ implies $u > 0$. To approximate its value we define the sequence $u_{j+1} = \mu(u_j)$, with $u_0 = r_1$. This sequence will converge toward u for large j. A proof of this statement is given in Appendix A.1.

In practice we can not deal evaluate infinite sums numerically, thus we need to choose a cutoff index K for the sums such that

$$\sum_{k=K+1}^{\infty} k\pi_k(u) \ll 1. \tag{1.59}$$ {?}

For scale-free network with exponent smaller than 2 for example, this sum always diverges and thus this method is not applicable.

Once u is approximated, we can compute the first K probabilities p_k, which is sufficient to sample random numbers between 1 and K with relative probability p_k. If K is chosen such that $r_k << 1$ for $k > K$, the degree distribution in the GCC closely approximate the distribution r_k.

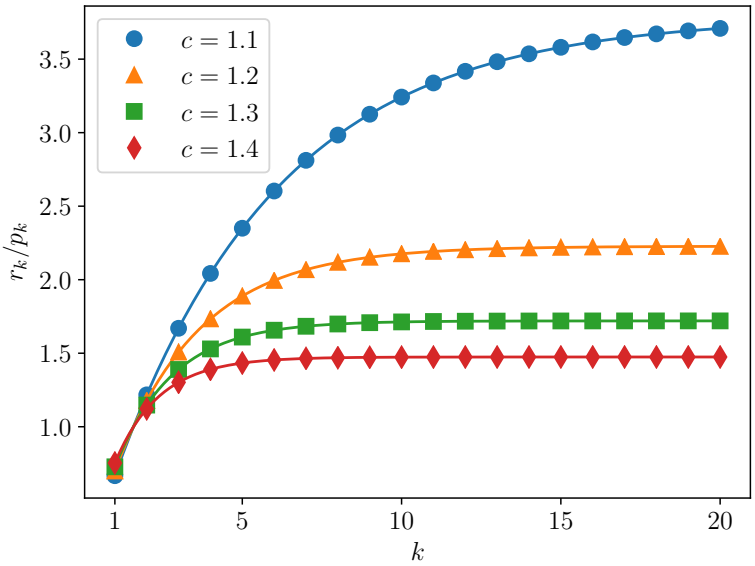
1.11.2 Erdos-Renyi reconstruction

In order to test the algorithm presented, we choose the target connected degree distribution r_k to be the degree distribution of the GCC of an Erdos-Renyi network. It is then expected that the reconstructed p_k closely approximate a Poisson degree distribution.

The probability p_k to have degree k in an Erdos-Renyi network is given in eq. (1.24). Using eq. (1.49) and (1.48) to find u and S yield everything we need to be able to determine the GCC degree distribution r_k from eq. (1.55). We can therefore use the algorithm on these r_k.

When computing S for the original Poisson distribution, we should however be cautious, as the reconstructed p_0 will always be 0. The expected resulting degree distribution, correctly normalized, is therefore

$$p_k = \frac{c^k}{k!}\frac{1}{e^c - 1}. \tag{1.60}$$ {?}



reconstruction⟩

FIGURE 1.9: Bias factor r_k/p_k for r_k being the degree distribution of the GCC of an Erdos-Renyi network with various mean degree c and cutoff constant $K = 10000$. The p_k have been determined using the algorithm presented in the text. Solid line is the expected value $(1 - u^k)/S$ for the bias factor.

The expected bias ratio r_k/p_k is shown for various mean degree c and a cutoff constant $K = 10000$ in fig. 1.9 together with the same value computed from the algorithm presented above. As it can be seen, the agreement is very good. During the computations it has been observed that the closer the mean degree is to the critical value $c = 1$, the slower the fixpoint iteration converges.

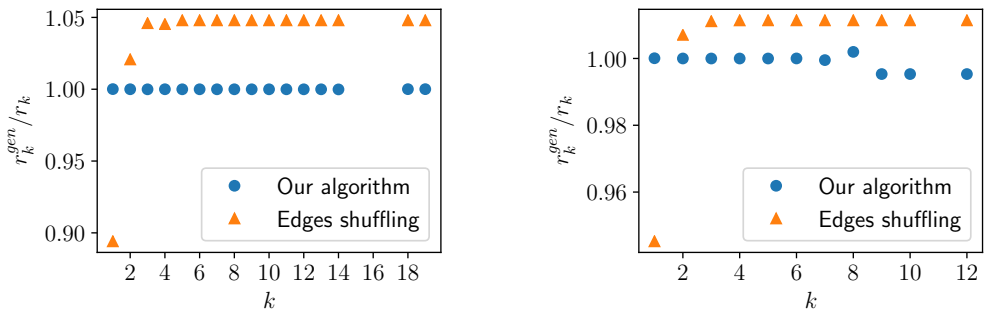
1.11.3 Real world networks

As an example of use of the algorithm presented, we apply it to real networks. We choose two by design connected network from the Konect network database [4], the powergrid of the western states of the United States and the road network of the state of California[4].

The connected degree distribution r_k is taken to be the empirical degree distribution of the real network considered. As a consequence, the cutoff constant K is the maximal degree appearing in the network. Then, to sample the resulting degree distribution p_k, we simply take the number of vertex d_k of degree k to be the closest integer to np_k, where n is the total number of nodes. In the examples presented we use $n = 10^7$.

We compare the degree distribution r_k^{gen} of the GCC of the newly generated network with the target distribution r_k by looking at the ratio r_k^{gen}/r_k. Resulting ratios are shown in fig. 1.10, together with the results obtained by taking the GCC of the reshuffled network.

[4]The codes of the networks in the Konect database are respectively UG and RO.



(A) (B)

gure: Real examples⟩

FIGURE 1.10: Plot of the ratio of the generated connected degree distribution r_k^{gen} to the target degree distribution r_k taken from a real network for our algorithm and the edges shuffling method. Missing values correspond to degree with probability 0 to appear. (A) Western US power-grid network. (B) California road network.

Chapter 2

Multiplex networks

tiplex networks⟩

2.1 Introduction

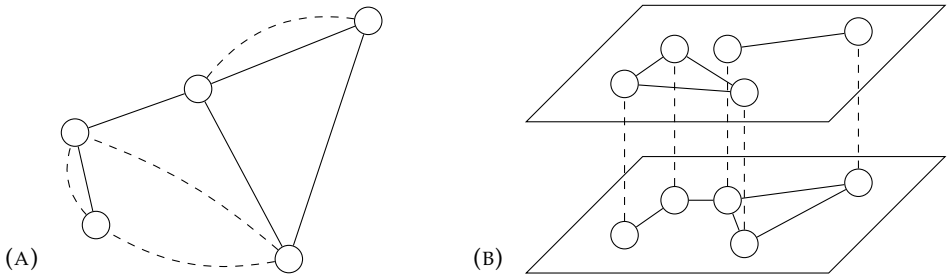
Move this to the general introduction ? Put the actual general introduction into the single layer network chapter ?

The concept of network can be extended by allowing different types of edges between the nodes of a network. Such generalized networks are called *multiplex networks* or *multi layers networks*. Real world example of such multiplex networks are numerous, for example there are many different types of connections between cities: transport connections (road, railroads, airlines), information connections (phone lines, internet connections) or supply connections (electricity, water).

Mathematically, it is convenient to represent multiplex networks in a slightly different but equivalent way. We consider each type of edge to generate its own network. If we started with L different type of edges, we therefore get L networks, all sharing the same set of nodes. Conceptually it is similar to stack L networks on top of each other, each network forming a *layer* of the composed multi layers network (hence the name). An illustration of both representation of multiplex networks is shown in fig. 2.1 for a two layers network.

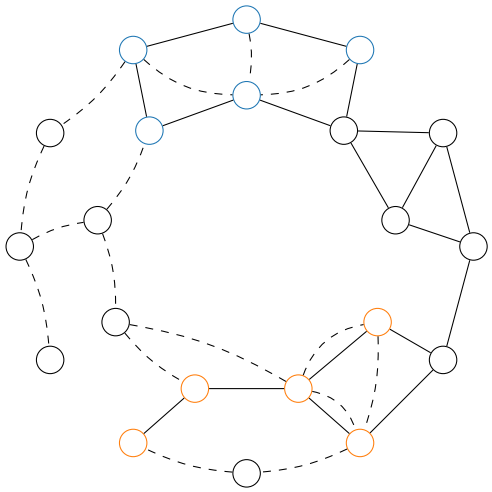
Thinking in term of layers allows to reuse all quantities define in the context of single layer networks, such as the generating functions. We the extended quantity the same way as the original one, adding a subscript i to its symbol to denote the fact that it applies to layer i, except if the symbol already has a subscript. In this case we add a superscript (i).

Finally, we only consider that each layer of a multiplex network is independant to each other. This is most likely not true for real networks, but we restrict ourself to this case for simplicity.



tiplex networks⟩

FIGURE 2.1: Possible representations of a two layers network. (A) A network with two kinds of edges, straight solid edges and curved dashed ones. (B) Two networks in separate layers with corresponding nodes.



ected is not viable⟩

FIGURE 2.2: Intersection of connected components of two layers of a multiplex networks, represented respectively with solid straight line and with curved dashed line. The intersection is composed of the colored nodes, however there is no path in the intersection connecting the blue and orange nodes. Therefore the intersection results in two distinct viable clusters.

2.2 Giant viable cluster

2.2.1 Size of the GVC

For single layer network, we are interested in the GCC as it is common for a system to be operational only if it is connected to the whole corresponding network. In the context of multiplex networks, the concept of connected component is extended by introducing *viable clusters*. A viable cluster is defined as a subset of the multiplex network in which there is a path between each pair of vertices in all layers. Note that the path between two vertices must be totally included in the viable cluster. As a consequence, the intersection of connected components is not in general a viable clusters as it is illustrated in fig. 2.2.

Correct the following explanation, it is going nowhere

The rationale behing this definition can be clarified by the introduciton of the *giant viable cluster* (GVC) which is the extension of the single layer concept of GCC, i.e. a viable cluster which occupy a non zero fraction of the whole network in the limit of large n. Let consider that a node is functional only if it is connected to the GVC. One example of such requirement is the network of powergrid regulation stations. Such stations must be connected to both the powergrid and an internal

nd a ref ?

Let $g_0^{(i)}$ and $g_1^{(i)}$ be the generating functions of respectively the degree and the excess degree distribution in layer i. Moreover define u_i as the probability that a vertex reached after following an edge in layer i is not part of the giant viable cluster. Be aware that u_i is not the direct extension of the single layer u quantity, which correspond to the probability not to be part of the GCC.

Then if we pick a vertex v at random the probability S that it is part of the giant viable cluster can be written as

$$S = P_0 \left(\bigcap_{i=1}^{L} \exists w \in N_i(v)\ w \in GVC \right). \tag{2.1}$$ {?}

By requiring that the layers are independent from one others, we can rewrite S as a product

$$S = \prod_{i=1}^{L} P_0^{(i)} \left(\exists w \in N_i(v)\ w \in GVC\right) \tag{2.2}$$ {?}

$$= \prod_{i=1}^{L} \left[1 - P_0^{(i)} \left(w \notin GVC\ \forall w \in N_i(v)\right)\right] \tag{2.3}$$ {?}

$$= \prod_{i=1}^{L} \left[1 - \sum_{k=0}^{\infty} P_0^{(i)} \left(w \notin GVC\ \forall w \in N_i(v) | \deg v = k\right) p_k^{(i)}\right] \tag{2.4}$$ {?}

$$= \prod_{i=1}^{L} \left[1 - \sum_{k=0}^{\infty} P_1^{(i)} \left(v \notin GVC | \deg v = k\right) p_k^{(i)}\right] \tag{2.5}$$ {?}

$$= \prod_{i=1}^{L} \left[1 - \sum_{k=0}^{\infty} u_i^k p_k^{(i)}\right] \tag{2.6}$$ {?}

$$= \prod_{i=1}^{L} \left[1 - g_0^{(i)}(u_i)\right]. \tag{2.7}$$ Multiplex GCC size

We can find u_j by a similar reasoning. First note that $1 - u_j$ is the probability that a vertex reached by following an edge in layer j is in the giant viable cluster. Which as before can be written in the form

$$1 - u_j = P_1^{(j)} \left(\bigcap_{i=1}^{L} \exists w \in N_i(v)\ w \in GVC\right) \tag{2.8}$$ {?}

$$= \prod_{i=1}^{L} P_1^{(j)} \left(\exists w \in N_i(v)\ w \in GVC\right). \tag{2.9}$$ {?}

Since the layers are independent, the fact that we reached v by following an edge in layer j to reach vertex v is irrelevant in all other layers. However in layer j this means that the degree distribution follows the distribution $q_k^{(j)}$ rather than $p_k^{(j)}$. Putting this together we get

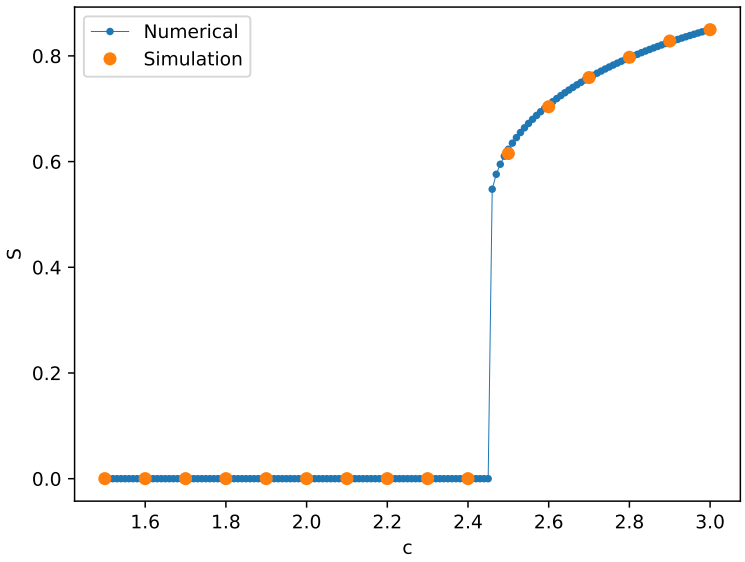
$$1 - u_j = \left[1 - \sum_{k=0}^{\infty} u_j^k q_k^{(j)}\right] \prod_{\substack{i=1 \\ i \neq j}}^{L} \left[1 - \sum_{k=0}^{\infty} u_i^k p_k^{(i)}\right] \tag{2.10}$$ {?}

$$= \left[1 - g_1^{(j)}(u_j)\right] \prod_{\substack{i=1 \\ i \neq j}}^{L} \left[1 - g_0^{(i)}(u_i)\right]. \tag{2.11}$$ Multiplex u final

2.2.2 Algorithm to find the viable clusters

Write. Make sure to introduce small components problem

(A)



(B)

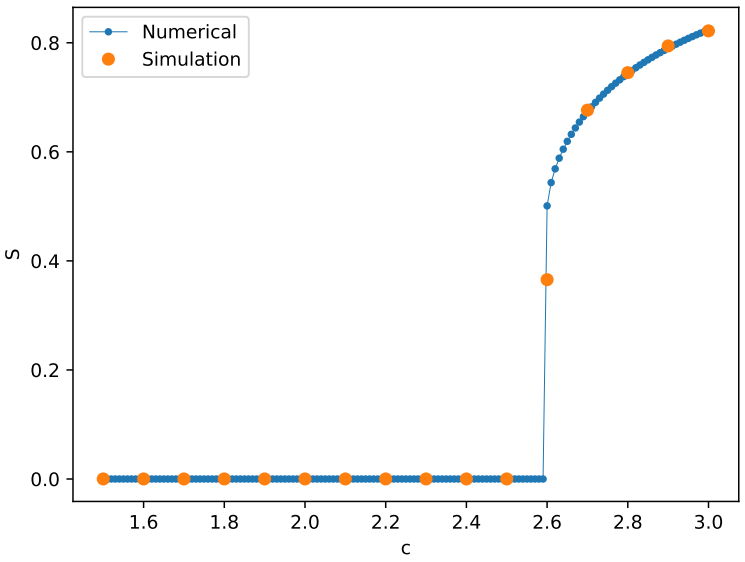


FIGURE 2.3: Numerical solution of eqs. (2.7) and (2.11), together with results on simulated networks for multiplex networks composed of two layers with the same distribution and mean number of edge c. Results of simulations are average over 10 runs and the network size was set to 10^4 nodes. Simulations ran for several seconds in total, indicating that much larger network should be easily usable. (A) Poisson degree distribution. (B) Geometric degree distribution.

2.3 Boundary condition

Write some introduction

Up to now, we have considered the multiplex generated to be determined via the degree distributions of each of its layer. However a degree distribution has an infinite number of degrees of freedom, therefore it is more practical to let the degree distributions depend on a finite set of parameters $\lambda_1, \lambda_2, \ldots, \lambda_N$ and express the behaviour of the network in term of them. Note that the number of parameters N does not need to match the number of layers L.

In order to make our main statement about the critical region for a multiplex network, we need to introduce several quantities. First, let introduce

$$\mathbf{u} = (u_1, u_2, \ldots, u_L) \tag{2.12}$$ {?}

$$\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_N) \tag{2.13}$$ {?}

$$f_j(\boldsymbol{\lambda}, \mathbf{u}) = 1 - u_j - \left[1 - g_1^{(j)}(u_j)\right] \prod_{\substack{i=1 \\ i \neq j}}^{L} \left[1 - g_0^{(i)}(u_i)\right] \tag{2.14}$$?Definition fj?

and the function

$$F : \mathbb{R}^N \times \mathcal{I}^L \to \mathbb{R}^L \tag{2.15}$$ {?}

$$(\boldsymbol{\lambda}, \mathbf{u}) \mapsto F(\boldsymbol{\lambda}, \mathbf{u}) = (f_1(\boldsymbol{\lambda}, \mathbf{u}), f_2(\boldsymbol{\lambda}, \mathbf{u}), \ldots, f_L(\boldsymbol{\lambda}, \mathbf{u})), \tag{2.16}$$?Definition F?

where $\mathcal{I} = [0, 1]$. The variables $\mathbf{u}$ in which we are interested are always in $\mathcal{I}^L$, since u_i represents a probability for all i.

Since the functions $g_0^{(i)}$ and $g_1^{(i)}$ are analytic with respect to the u_i, the function

$$F_{\boldsymbol{\lambda}} : \mathcal{I}^L \to \mathbb{R}^L \tag{2.17}$$ {?}

$$\mathbf{u} \mapsto F_{\boldsymbol{\lambda}}(\mathbf{u}) = F(\boldsymbol{\lambda}, \mathbf{u}), \tag{2.18}$$ {?}

is continuously differentiable for all parameters $\boldsymbol{\lambda}$. Therefore we can define Jacobi matrix $J(\boldsymbol{\lambda}, \mathbf{u})$ of $F_{\boldsymbol{\lambda}}$ as having coefficients

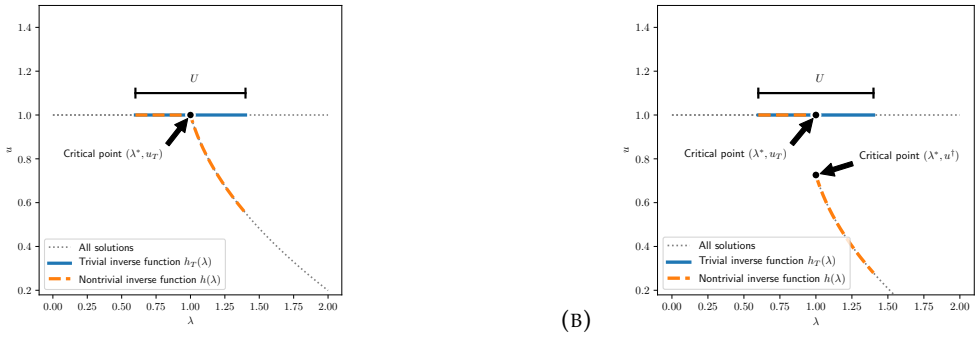
$$[J(\boldsymbol{\lambda}, \mathbf{u})]_{ij} = \frac{\partial f_i(\boldsymbol{\lambda}, \mathbf{u})}{\partial u_j}. \tag{2.19}$$ {?}

With the help of the notation introduced, we can now express solving eq. (2.11) as being equivalent to finding $\mathbf{u}^* \in \mathcal{I}^L$ such that

$$F(\boldsymbol{\lambda}, \mathbf{u}^*) = 0. \tag{2.20}$$ Implicit equation

If this equation only admits the trivial solution $\mathbf{u}^* = \mathbf{u}_T$, the parameter $\boldsymbol{\lambda}$ corresponds to a state without GVC. On the other if multiple solutions $\mathbf{u}^*$ exist, a GVC must exist as well. To determine the boundary between these two regions (i.e. the critical region), we use the implicit function theorem.

First, we assume that F (and not only $F_{\boldsymbol{\lambda}}$) is continuously differentiable and that we know a solution $\mathbf{u}^*$ of (2.20) for some parameter vector $\boldsymbol{\lambda}^*$. With that assumption the implicit function theorem can be state as follow:



(A) (B)

s phase transitions)?

FIGURE 2.4: (a) Scheme of a continuous phase transition. (b) Scheme of a discontinuous phase transition.

If $\det\left[J(\boldsymbol{\lambda}^*, \mathbf{u}^*)\right] \neq 0$ then there is an open neighbourhood $U \subset \mathbb{R}^L$ of $\boldsymbol{\lambda}^*$ such that there is an unique continuously differentiable function $h : U \to \mathcal{I}^L$ with

$$h(\boldsymbol{\lambda}^*) = \mathbf{u}^* \tag{2.21}$$

$$F(\boldsymbol{\lambda}, h(\boldsymbol{\lambda})) = 0, \quad \forall \boldsymbol{\lambda} \in U. \tag{2.22}$$

Implicit soluti

The result in which we are interested here comes from the contrapositive of this statement, namely that if for all neighbourhoods U we can not find a uniquely defined continuous function h, then the determinant of the Jacobi matrix $J(\boldsymbol{\lambda}, \mathbf{u})$ must be zero,

$$\det\left[J(\boldsymbol{\lambda}^*, \mathbf{u}^*)\right] = 0. \tag{2.23}$$

Boundary condit

This condition has previously been outlined, without proof in [5]. We now prove that such situations arise if $\boldsymbol{\lambda}^*$ is a critical point of the phase transition between the absence and existence of a GVC, and therefore that eq. (2.23) is a sufficient condition to find the critical region of such phase transition.

First notice that in the context of multiplex network a phase transition appears between the trivial solution $\mathbf{u}_T = (1, 1, \ldots, 1)$ (where $S = 0$) and non trivial solutions ($S > 0$). However, the trivial solution $\mathbf{u}_T$ solves eq. (2.11) for any generating function and thus for any parameter vector $\boldsymbol{\lambda}$. For a continuous phase transition this immediately gives us $\mathbf{u}^* = \mathbf{u}_T$. Moreover, on one side of the phase transition occurring at $\boldsymbol{\lambda}^*$ one solution exists, while on the other at least two do. Therefore for any U open containing $\boldsymbol{\lambda}^*$ we can define two distinct functions on U that fulfil eq. (2.22), the trivial $h_T(\boldsymbol{\lambda}) = \mathbf{u}_T$ and another function h corresponding to the non trivial solutions, with $h(\boldsymbol{\lambda}^*) = h_T(\boldsymbol{\lambda}^*) = \mathbf{u}_T$. So the function h of the implicit function theorem is not uniquely defined and thus $\det\left[J(\boldsymbol{\lambda}^*, \mathbf{u}_T)\right] = 0$.

On the other hand, let consider a discontinuous phase transition at $\boldsymbol{\lambda}^*$. For any neighbourhood U of $\boldsymbol{\lambda}^*$ there are two sequences $(\boldsymbol{\lambda}_n, \mathbf{u}_n)$ and $(\boldsymbol{\eta}_m, \mathbf{v}_m)$ with $\boldsymbol{\lambda}_n, \boldsymbol{\eta}_m \in U$ such that

$$\lim_{n\to\infty} (\boldsymbol{\lambda}_n, \mathbf{u}_n) = (\boldsymbol{\lambda}^*, \mathbf{u}^\dagger) \quad \text{avec } \mathbf{u}^\dagger \neq \mathbf{u}_T \tag{2.24}$$

$$\lim_{m\to\infty} (\boldsymbol{\eta}_m, \mathbf{v}_m) = (\boldsymbol{\lambda}^*, \mathbf{u}_T) \tag{2.25}$$

$$F(\boldsymbol{\lambda}_n, \mathbf{u}_n) = 0 \quad \forall n \tag{2.26}$$

$$F(\boldsymbol{\eta}_m, \mathbf{v}_m) = 0 \quad \forall m. \tag{2.27}$$

If we assume that an unique continuous function h solving eq. (2.22) exists, we would have

$$h(\boldsymbol{\lambda}_n) = \mathbf{u}_n \quad \forall n \tag{2.28}$$ {?}

$$h(\boldsymbol{\eta}_m) = \mathbf{v}_m \quad \forall m. \tag{2.29}$$ {?}

The continuity of h would furthermore imply

$$h(\boldsymbol{\lambda}^*) = \lim_{n\to\infty} h(\boldsymbol{\lambda}_n) = \lim_{n\to\infty} \mathbf{u}_n = \mathbf{u}^\dagger, \tag{2.30}$$ {?}

but also

$$h(\boldsymbol{\lambda}^*) = \lim_{m\to\infty} h(\boldsymbol{\eta}_m) = \lim_{m\to\infty} \mathbf{v}_m = \mathbf{u}_T. \tag{2.31}$$ {?}

Since $\mathbf{u}_T \neq \mathbf{u}^\dagger$, this gives raise to the contradiction $h(\boldsymbol{\lambda}^*) \neq h(\boldsymbol{\lambda}^*)$. Therefore our assumption must be false and no continuous function h can be defined to solve eq. (2.22). So finally, we have $\det\left[J(\boldsymbol{\lambda}^*, \mathbf{u}^*)\right] = 0$, $\mathbf{u}^*$ being either $\mathbf{u}_T$ or $\mathbf{u}^\dagger$.

If $L = N = 1$, the problem simplifies to the single layer case in which the degree distribution is determined by a single parameter λ. In that case the Jacobi matrix J reduces to the scalar quantity

$$J(\lambda, u) = \frac{\partial}{\partial u}\left(g_1(u) - u\right) = \frac{\partial g_1(u)}{\partial u} - 1. \tag{2.32}$$ {?}

Therefore the condition for the boundary $\det J(\lambda, u) = 0$ becomes

$$\frac{\partial g_1(u)}{\partial u} = 1. \tag{2.33}$$ {?}

This condition correctly reduces to the one given previously in eq. (1.50).

2.4 Interval estimation of the critical region

2.4.1 Motivation

In order to verify that eq. (2.23) indeed gives the critical region for a multiplex network, we need to introduce an independent numerical method to approximate it. This comes down to find the number of solutions of eq. (2.11) in the whole parameter space and then draw the boundary of the regions with a constant number of solution. However this leads to two problems.

First, standard algorithms to find zeros of functions do not guarantee that all zeros are found. These algorithms usually found one solution at a time, the one being found depending on the initial guess solution provided by the user. This causes immediate problem for our purpose, since missing a solution of eq. (2.11) may drastically change our estimation of the boundary region.

Secondly, the standard algorithms can only deal with one point of the parameter space at a time. This forces us to restrict our search on a discrete set of parameters. If this set is ill chosen, we may miss important features of the critical region.

The second point is minor compare to the first. We indeed expect the critical region to be smooth and therefore restricting our analysis on any lattice should reasonably approximate it. We still mention this caveat however, since it can be solved

by the same method as the first one, namely by introducing *interval arithmetic*. Interval arithmetic is a framework, presented in the next section, that allows to guarantee under some circumstance that all zeros of a function have been found and to work with regions of the parameter space rather than discrete points.

2.4.2 Interval arithmetic

Let $C \subset \mathcal{R}^L$ be the set of all parameters $\boldsymbol{\lambda}$ corresponding to a critical point. This set correspond to the parameters that solve simultaneously eq. (2.11) and eq. (2.23) for some $\mathbf{u} \in \mathcal{I}^L$.

First of all an *interval* I is defined a set of the form

$$I = [a, b] = \{x \in \mathbb{R} | a \leq x \leq b\}. \tag{2.34}$$ {?}

The set of all intervals is denoted as $\mathbb{IR}$. The N-dimensional equivalent of an interval is an *interval box* B, defined as the Cartesian product of N intervals,

$$B = I_1 \times I_2 \times \cdots \times I_N, \quad I_k \in \mathbb{IR} \quad \forall k = 1, \ldots, N \tag{2.35}$$ {?}

The set of all N-dimensional interval boxes is denoted $\mathbb{IR}^N$.

Given a function $\phi : \mathbb{R}^M \to \mathbb{R}^N$ we define a new interval valued function $\Phi : \mathbb{IR}^M \to \mathbb{IR}^N$ such that

$$x \in B \quad \Rightarrow \quad \phi(x) \in \Phi(B). \tag{2.36}$$ Definition inte

A function Φ with this property is called an *interval extension* of ϕ. In other words $\Phi(B)$ is guaranteed to contains all possible image of B under ϕ. Note that an interval extension is not unique, and we would like to use the *tightest* possible extension, meaning that we want $\Phi(B)$ to approximate $\phi(B)$ as closely as possible. A perfect estimation is however not possible in general since the image of an interval box is not always itself an interval box.

Note that as oppose to ϕ, its interval extension Φ is exactly representable numerically. This can be done by requiring the fact that the implementation of Φ includes numerical inaccuracy in such a way that the eq. (2.36) holds for the finite precision numerical result $\Phi(B)$. Hopefully good implementations, that respect this condition and are reasonably tight, can be found in existing libraries. In this thesis we use the implementation provided by the Julia package `IntervalArithmetic.jl`[?].

sing ref

The next step is to solve eq. (2.11). Several general schemes exist to solve equations in a guaranteed way using interval arithmetic[?], but here we use a simpler algorithm inspired by them and more suited to our present needs.

sing ref

We define

$$\psi(\boldsymbol{\lambda}, \mathbf{u}) = F(\boldsymbol{\lambda}, \mathbf{u}) + \mathbf{u}, \tag{2.37}$$ {?}

with components

$$\psi_j(\boldsymbol{\lambda}, \mathbf{u}) = 1 - \left[1 - g_1^{(j)}(u_j)\right] \prod_{\substack{i=1 \\ i \neq j}}^{L} \left[1 - g_0^{(i)}(u_i)\right]. \tag{2.38}$$ {?}

Also we define Ψ as an interval extension of ψ.

Now, let $\mathbf{u}^* \in U_0$ be a solution of (2.20) for some $\boldsymbol{\lambda} \in \Lambda$. By the definition of ψ and eq. (2.36),

$$F(\boldsymbol{\lambda}^*, \mathbf{u}^*) = 0 \quad \Rightarrow \quad \mathbf{u}^* = \psi(\boldsymbol{\lambda}^*, \mathbf{u}^*) \in \Psi(\Lambda, U_0). \tag{2.39}$$ {?}

Therefore if we apply $\Psi(\Lambda, \cdot)$ to an interval box U_0 containing a solution, the resulting interval box contains the solution as well. We know that all solutions $\mathbf{u}^*$ are contained in $\mathcal{I}$ by definition of $\mathbf{u}$ and thus by iterating the previous argument from $U_0 = \mathcal{I}$, all solutions are elements of the interval boxes $U_k(\Lambda)$ recursively defined by

$$U_{k+1}(\Lambda) = \Psi(\Lambda, U_k(\Lambda)). \tag{2.40}$$ Recursion relation

Therefore if we find k such that $U_k = \{\mathbf{u}_T\}$, we know that the system for any $\boldsymbol{\lambda} \in \Lambda$ only admits the trivial solution.

In practice however, the sequence U_k never converges to exactly the set $\{\mathbf{u}_T\}$, we therefore consider the criterion to be met if

$$U_k \subset [1 - \varepsilon, 1]^L, \tag{2.41}$$ Criterion for trivi

for some small tolerance ε.

Furthermore it is possible in some cases to guarantee the presence of non trivial solutions, allowing to conclude that a GVC emerges. Indeed, if we can find some interval boxes Λ and U such that

$$\Psi(\Lambda, U) \subset U \quad \text{and} \quad \mathbf{u}_T \notin U, \tag{2.42}$$ Criterion for non t

then by definition of the interval extension (eq. (2.36)), we have

$$\psi(\boldsymbol{\lambda}, U) \subset U, \quad \forall \boldsymbol{\lambda} \in \Lambda. \tag{2.43}$$ {?}

Since U is closed and simply connected, the fixpoint theorem [?] applies, implying that for each $\boldsymbol{\lambda} \in \Lambda$ there must be at least one $\mathbf{u}^*$ in U such that $\mathbf{u}^*$ is a fixpoint, or in other words such that $\mathbf{u}^* = \psi(\boldsymbol{\lambda}, \mathbf{u}^*)$. Therefore eq. (2.42) is a sufficient condition to prove the existence of at least one solution. Moreover since we imposed $\mathbf{u}_T \notin U$, the solution present can not be the trivial one.

Find back which on exactly

Missing ref

2.4.3 Algorithm

Equations (2.41) and (2.42) give guaranteed criteria for respectively the absence and presence of a non trivial solution in the region Λ considered. This is sufficient to propose an algorithm to estimate the critical region C.

1. Choose an initial parameter region and store it in the set $\mathcal{S}_{\text{working}}$ of regions yet to be processed.
2. If $\mathcal{S}_{\text{working}}$ is empty terminate, otherwise retrieve the next parameter region from $\mathcal{S}_{\text{working}}$, and call it Λ.
3. If the radius of Λ is smaller than some tolerance δ, store it in the set $\mathcal{S}_{\text{unkown}}$ of regions for which the algorithm is unable to conclude using the tolerance δ.
4. Compute $U_k(\Lambda)$ for k big, using eq. (2.40).
5. If $U_k(\Lambda)$ fulfil eq. (2.41), store Λ in the set $\mathcal{S}_{\text{trivial}}$ of trivial regions and go to 2.

6. Take a subset V of $U_k(\Lambda)$ such that $\mathbf{u}_T \notin V$.
7. If V fulfil eq. (2.42), store Λ in the set $\mathcal{S}_{\text{GVC}}$ of non trivial regions and go to 2.
8. Bisect Λ in two sub regions and add both to $\mathcal{S}_{\text{working}}$. Go to 2.

By construction, at any step the critical region is contain in the union of the intervals with unknown status, i.e.

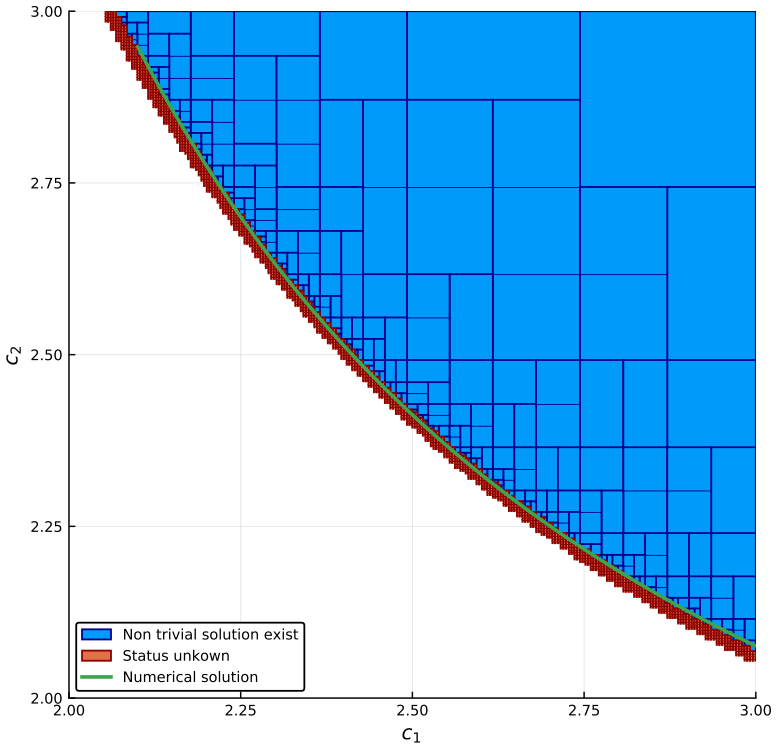
$$C \subset \bigcup_{U \in \mathcal{S}_{\text{unkown}}} U. \tag{2.44}$$ {?}

Thus when the algorithm terminates, this union gives an estimation of C.

2.5 Results

We apply the two methods presented to different cases of multiplex network in order to test it. The algorithm using interval arithmetic is compared to the numerical solution of the system composed by eq. (2.11) and (2.23), computed using the `NLsolve.jl` Julia package.

Choose what multiplex networks should be used and with what parameters and actually produce the results.



s and boundary⟩?

Make the plot more readable and less ugly

Find why the two methods seems drift away one from the other far from the center.

FIGURE 2.5: Phase diagram for a multiplex network composed of two Erdos-Renyi layer with mean degree c_1 and c_2. In the blue region a non trivial solution for $\mathbf{u}$ has been found, in the uncolored region only the trivial solution $\mathbf{u}_T$ exists and in the red region the algorithm was unable to conclude in favor of either case. The solid green line is the numerical solution to eq. (2.11) and (2.23).

Appendix A

Fixpoint iteration for connected networks generation

A.1 Convergence of the fixpoint iteration

int convergence⟩ First notice that the case $r_1 = 0$ is trivial, as described in the main text. We will therefore assume in this Appendix that $r_1 > 0$, immediately giving $\mu(0) = r_1 > 0$. Second, not that (1.58) tells us that $z < 1$ implies $\mu(z) < 1$. From there we separate two cases:

If $u = 1$ is the unique solution of eq. (1.57) then $\mu(z)$ must be continuous for $z \in [0, 1]$ and $\mu'(1) < 1$, making $u = 1$ and attractive fixpoint. On the other hand if there is another solution u^* to eq. (1.57), it is the unique solution with $0 \leq u^* < 1$ since $\mu(z)$ is an increasing function of z, as it is demonstrated in Appendix A.2. Moreover, since $\mu(0) > 0$ we have $\mu'(u^*) < 1$, which makes it an attracting fixpoint and makes $u = 1$ a repulsive one.

We can thus conclude that the fixpoint iteration proposed always converges and converges to the degenerate case $u = 1$ only if it is the unique possibility.

A.2 Monotonicity of $\mu(z)$

x: Monotonicity⟩ To prove that $\mu(z)$ is an increasing function, we compute its derivative with respect to z, which yields

$$\mu'(z) = \left[\sum_{k=1}^{\infty} k\pi_k(z)\right]^{-2} (s_1(z) + s_2(z)) \tag{A.1}$$ {?}

$$s_1(z) = \sum_{j,k} kj\pi'_k(z)\pi_j(z)\left(z^{k-1} - z^{j-1}\right) \tag{A.2}$$ {?}

$$s_2(z) = \sum_{j,k} k(k-1)j\pi_k(z)\pi_j(z)z^{k-2}. \tag{A.3}$$ {?}

The sum $s_1(z)$ can be rewritten as

$$s_1(z) = \sum_{j>k} kj\left(\pi'_k(z)\pi_j(z) - \pi'_j(z)\pi_k(z)\right)\left(z^{k-1} - z^{j-1}\right) \tag{A.4}$$ {?}

$$= \sum_{j>k} \frac{kr_k}{1-z^k}\frac{jr_j}{1-z^j}\frac{z^k - z^j}{z^2}\left(\frac{k}{z^{-k}-1} - \frac{j}{z^{-j}-1}\right) \tag{A.5}$$ {?}

$$= \sum_{j>k} kj\pi_k(z)\pi_j(z)\frac{z^k - z^j}{z^2}\left(\frac{k}{z^{-k}-1} - \frac{j}{z^{-j}-1}\right). \tag{A.6}$$ {?}

Using the fact that the function

$$f_z(\lambda) = \frac{\lambda}{z^{-\lambda} - 1} \tag{A.7}$$ {?}

is a decreasing function of λ we can see that for $z \in [0, 1)$ and $j > k$ we have

$$z^k - z^j \geq 0 \tag{A.8}$$ {?}

$$\frac{k}{z^{-k} - 1} - \frac{j}{z^{-j} - 1} \geq 0, \tag{A.9}$$ {?}

and thus $s_1(z) \geq 0$. Moreover each terms in $s_2(z)$ is non-negative, so we have $s_2(z) \geq 0$. We can therefore conclude that $\mu'(z) \geq 0$ and thus that $\mu(z)$ is an increasing function of z.

Bibliography

[1] M. Newman. *Networks: an introduction*. 2010.

[2] M. Bauer and D. Bernard. "Maximal entropy random networks with given degree distribution". In: *arXiv preprint cond-mat/0206150* (2002).

[3] P. Bialas and A. K. Oleś. "Correlations in connected random graphs". In: *Physical Review E* 77.3 (2008), p. 036124.

[4] J. Kunegis. "Konect: the koblenz network collection". In: *Proceedings of the 22nd International Conference on World Wide Web*. ACM. 2013, pp. 1343–1350.

[5] G. Baxter et al. "Avalanche collapse of interdependent networks". In: *Physical review letters* 109.24 (2012), p. 248701.