MODELING COMPLEX SYSTEMS BY EVOLVING NETWORKS

Doctoral Dissertation, 2007

GÁBOR CSÁRDI

Department of Biophysics, KFKI Research Institute for
Particle and Nuclear Physics, Hungarian Academy of Sciences,
Budapest, Hungary

Advisor: Dr. Péter Érdi, Ph.D., D.Sc.,
Henry R. Luce Professor, Head of Department

Faculty of Informatics, Eötvös Loránd University,
Budapest, Hungary

Ph.D. School: Informatics
Head: Dr. János Demetrovics
Ph.D. Program: Information Systems
Head: Dr. András Benczúr

It is that the sciences do not try to explain, they hardly even try to interpret, they mainly make models. By a model is meant a mathematical construct which, with the addition of certain verbal interpretations, describes observed phenomena. The justification of such a mathematical construct is solely and precisely that it is expected to work—that is, correctly to describe phenomena from a reasonably wide area. Furthermore, it must satisfy certain esthetic criteria—that is, in relation to how much it describes, it must be rather simple.

John von Neumann, Method in the Physical Sciences

Acknowledgements

I would like to thank my advisor, Péter Érdi who has been guiding my work for about six years now, sometimes in Budapest, sometimes in Kalamazoo, for teaching me more disciplines than my brain was actually capable of taking in. Péter, I know it seems impossible, but you have good chance of improving the score in our endless ping-pong battle.

I would like to thank my colleagues, László Zalányi, Tamás Kiss, Máté Lengyel, whom I worked with on the projects presented in this dissertation. I should mention here that the contents of Sections 2.2.2.2 and 2.2.2.3 are solely their work, I did not significantly contributed to the results. Leaving out these sections would distort the project, leaving out the project from the dissertation would hurt my good memories, so they are included.

I enjoyed the meetings with my collaborators, Jan Tobochnik and Katherine Strandburg, I couldn't had written this dissertation without their valuable input. Thanks to the students I worked with, Daniel Catlin, Andrew Schneider, Elliot Paquette and Hannah Masuga.

In 2000 it seemed a good idea joining the computational neuroscience group at KFKI, I am grateful to all members of the group. Thanks to Tamás Nepusz for the work we did on igraph.

Thanks to my family and Nadia for motivation and their patience.

Contents

List of Figures

List of Tables

1 Introduction

THE WORK PRESENTED in this dissertation uses the tools of graph theory. It is thus appropriate to—at least briefly—introduce this discipline.

The graph model means the description by binary relations. The idea is basic, yet universal. If we think of the parent child relationship in a family tree, the "connected by a flight" relationship in the airline network, the wires connecting Internet routers, the links from one web page to another, these are just a few examples to be grasped by the (binary) graph model. A gene regulates the expression of another gene, a predator eats its prey, the Reader might call his or her friend on his or her cell phone, or just send an email to him or her. Clearly, binary relationships are everywhere.

The trial of a very general mathematical model is its utility. It can either tell something new that other models can't and will rise, or it will fail. Although the author does not dare to ask the Reader to make the judgement merely on the contents of this dissertation, he cannot deny hoping that his work serves as a (maybe subtle) evidence in the defense of graph theory and network science.

1.1 Classic graph theory

We do not intend to include all basic graph theory definitions normally appearing in the first chapter in any of the numerous graph theory books, but focus on the ones needed in the following chapters.

There are many definitions of graphs in the literature, equivalent for most purposes. Here is one of them [Andrásfai, 1997]. A *graph* is a triple,

$G = (V, E, \mathcal{G})$, where V and E are disjunct sets and $\mathcal{G}$ maps E to pairs of V. If the pairs are ordered then the graph is called *directed*, otherwise it is *undirected*. The elements of V are the *vertices* (or nodes) of the graph, the elements of E are the *edges* of the graph.

If $v, w \in V$ and $\mathcal{G}$ maps e to the (v, w) ordered pair then we say that e is an *outgoing edge* of v and an *incoming edge* of w; we also say that v and w are *neighbors* and both the v and e pair and the w and e pair are *adjacent*. If $v = w$ then e is called a *loop edge*, if there is also an $f \in E$ to which $\mathcal{G}$ maps (v, w) then we call e and f *multiple edges*. A graph without multiple edges and loop edges is called a *simple graph*. The *degree* of a vertex is the number of adjacent edges to it. The *in-degree* is the number of incoming adjacent edges, the *out-degree* is the number of outgoing adjacent edges. Naturally, the definitions of this paragraph can be defined for undirected graphs as well.

A graph is called an *empty graph* if E is the empty set. A simple graph is called a *full graph* if $\mathcal{G}$ maps to all possible (ordered or not ordered) non-loop pairs of vertices.

Two graphs are called *isomorphic* if there is a bijective relation between their vertices and edges, which also keeps the connections. For many purposes two isomorphic graphs are identical.

For each graph, a set of points in 3D Euclidean space (T) can be defined, which we call the *geometric realization* of the graph. We do not bore the reader with the actual definition, just mention that the vertices correspond to certain points of T and two vertices are connected with a (not necessarily straight) line if there is an edge in the graph which maps to the two vertices. For directed graphs the lines also have a direction. Fig. 1.1 shows the geometric realizations of some graphs.

A *path* is a sequence edges $\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}$, we say that the path goes from v_1 to v_n. If $v_1 = v_n$ then the path is a *circle*. (Obviously, the same can be defined for directed graphs as well.) A graph in which there is a path from every vertex to every other vertex is called a *connected* graph. The *shortest path* from a vertex to another is one containing minimum number of edges (the shortest path is not generally unique). If a graph is not connected then is has multiple *components*, a component is a maximal set of vertices in which there is a path from every vertex to every other vertex. Components are sometimes called *clusters*.

The *transitivity* of a network is defined as the number of connected triangles divided by the number of connected triples, an additional constant factor scales the value to $[0, 1]$:

$$C = 3 \cdot \frac{\text{number of connected triangles}}{\text{number of connected triples}}. \tag{1.1}$$

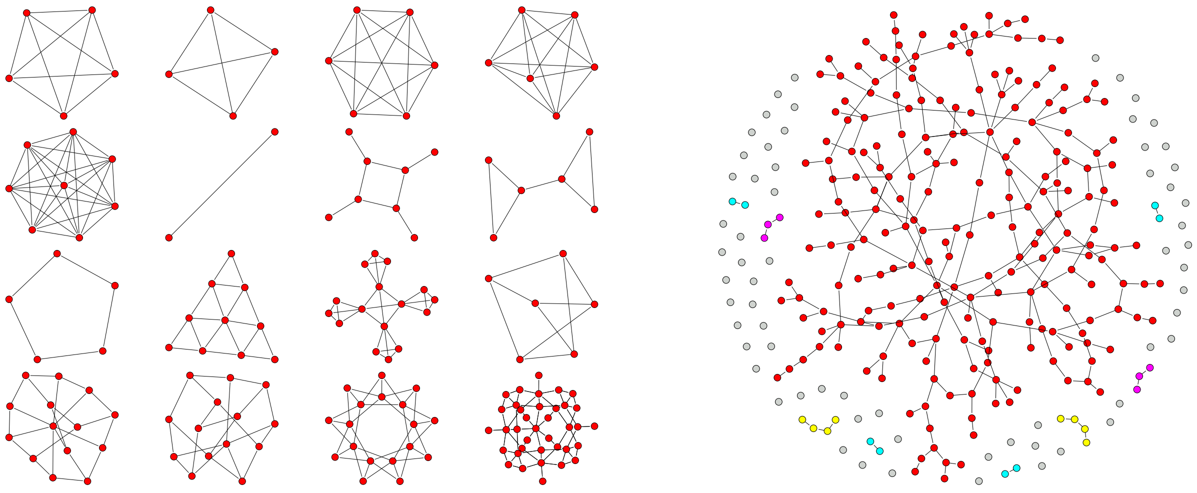


Figure 1.1: A couple of regular graphs and a random graph. (More precisely, the geometric realizations of some graphs.)

See [Newman, 2003a] for the definition of local transitivity and the difference between the two transitivity definitions.

We will use attributed graphs later on. An *attributed graph* is a tuple $(V, E, \mathcal{G}, S, \mathcal{A})$, where $(V, E, \mathcal{G})$ is a graph and $\mathcal{A}$ is a function mapping vertices into the S non-empty set. In general, we do not assume anything about the S set, it is not necessarily even ordered.

1.1.1 Euler's problem

There is no introduction to graph theory without Euler's problem, so let's get over it. The first documented graph theoretical work is from 1736, by Leonhard Euler. The town of Königsberg (look for Kaliningrad on the map, nowadays) is run across by the Pregel river. Seven bridges crossed the river in Euler's time, see Fig. 1.2.

The citizens posed the question whether it is possible to take a walk and cross every bridge exactly once. Euler proved that there is no such walk by proving that every vertex except for (at most) two must have an even vertex degree in a connected graph to make such a walk possible. In the Königsberg graph all vertex degrees are odd.

Classic graph theory has its classic questions, some of them already solved, others still open. Graph coloring, planar graphs, network flows, many of them would deserve a couple of pages here and it is to be feared that they turned out to be (much) more interesting than the main topic of the dissertation,

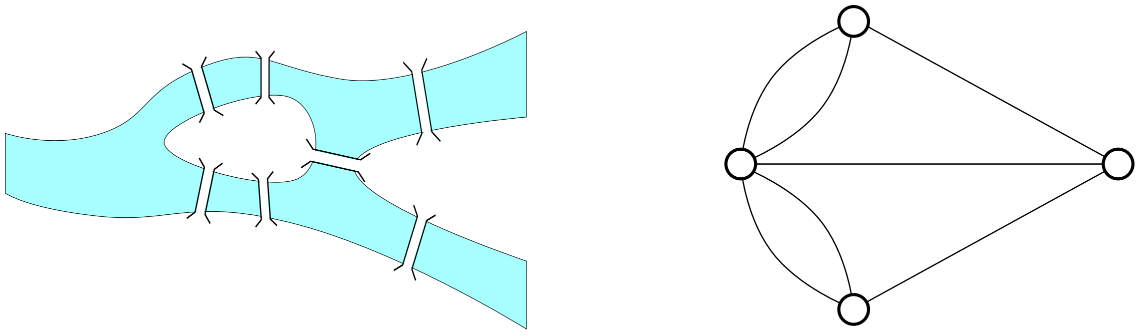


Figure 1.2: The bridges of Königsberg. A schematic drawing and an even simpler graph model.

so we leave them out. There are many excellent graph theory books, see e.g. [Bollobás, 2004, Diestel, 2006, Gross, 2005, Harary, 1994]. See [Wasserman and Faust, 1994] for an extensive textbook on social network analysis.

1.2 The statistical approach

We show three representative models from the recent developments of statistical graph theory, also called network science.

1.2.1 Random graphs

Early graph theory was restricted almost solely to the study of regular structures. Random graphs were first defined by Solomonoff and Rapoport [1951] and then rediscovered by Erdős and Rényi [1959, 1960, 1961]. They were the first to focus on statistical properties of a distribution over an ensemble of graphs instead of a given structure. They defined the most basic random graph models, the $G_{n,p}$ and the $G_{n,m}$ models. Here we briefly discuss $G_{n,p}$, the other model, $G_{n,m}$ has very similar properties.

$G_{n,p}$ is an ensemble of graphs with n vertices, where the probability of a graph is defined by the p parameter: it is the probability that an edge is present in the graph, independently of the other possible edges. In other words, $G_{n,p}$ is defined by an algorithm: take n vertices and for every pair make a connection with probability p. Clearly, the procedure can create every simple graph with n vertices (if $0 < p < 1$), but not with the same probability. Erdős and Rényi asked questions about the probability that an ensemble of $G_{n,p}$ graphs have a given statistical property or not. Notable examples are the degree distribution of these graphs and the presence of a giant component.

They took the limit of the infinite network, an ingredient of statistical physics. They increased n while keeping $z = (n-1)p$ constant. The degree distribution of the limit network is a Poisson distribution around the mean degree z, meaning that large vertex degree has extremely small probability.

A random graph is said to have a giant component if, again in the limit of the infinite graph and keeping the mean degree constant, the largest connected component of it covers an infinite number of vertices. They concluded that the graph has a giant component if $z > 1$, but not if $z < 1$. The $z = 1$ point is the place of a phase transition, and the average size of components diverges here.

See the book by Bollobás [1985] for (much) more on random graphs. See the works by Newman et al. [2001] and Molloy and Reed [1998] for a generalization of random graphs.

1.2.2 Small worlds

Everyone has the experience of sitting next to a complete stranger on the train, who turns out to work at the same place as the wife of our brother, or who went to the same school as our parents. We live in a small world. This has been actually shown by many studies, the first famous one was conducted by Milgram [1967]. Based on experiments with sending a packet from one end of the United States to another, by using only people's close acquaintances, he estimated that the average number of handshakes needed to get to any person from another one is about six. Six degrees of separation. Anyone to anyone else in the world. (The results were generalized to whole world seamlessly.)

This leads to the small world problem: how is this possible? It is well known that social networks (like the one Milgram utilized in his experiment) are highly transitive: the friends of my friends tend to be my friends as well. But how can the average longest distance in the network, the property Milgram estimated, be so small, six for the social network of the whole population of the world?

Watts and Strogatz [1998] gave a very simple model which could easily reproduce the small world phenomenon. Place the vertices on a regular lattice and connect all vertices closer to each other than a small threshold, like five steps. These connections represent our regular, highly transitive relationships. Then take some connections and rewire them completely randomly. In other words, an end point of an edge is rewired with probability p to a uniformly randomly chosen new vertex, p is a parameter of the model. These random connections represent our "random" meetings with strangers who were originally not part of our small social circle. That's it. If the value of p is favorable, then, again, in the limit of an infinite network, the average distance will be logarithmically small compared to the size of the network because of the random connections, and the transitivity of the graph will be high because of most of the original transitive connections are kept.

Small world networks lie somewhere between the order and randomness. If $p = 0$ then no rewiring takes place and we stay with the completely regular, ordered lattice. High transitivity, thousand (a million?) degrees of separation. If $p = 1$ then every edge is rewired, we essentially get a $G_{n,m}$ random network. Small transitivity, very (logarithmically) short distances. In between we can have both. Yes, our friends' friends are out friends most of the time, but sometimes not, and that is enough.

1.2.3 Scale-free graphs

It is an important question how the degree of the vertices in a given network is distributed. Do most vertices have the same degree, or is there a big variance? In case of vaccination against a sexually transmitted disease it is crucial to know whether most people have the same number of partners or there are some *hubs* with much more connections than the rest [Liljeros et al., 2001].

We know from the work of Price [1965] that the (in-)degree distribution of scientific citation networks is usually a power-law: most papers are never cited, or just a couple of times, whereas a few of them all the time [Mitzenmacher, 2004, Newman, 2005]. Price [1976] also gave a minimal model to explain this phenomenon, later the model was named preferential attachment by Barabási and Albert [1999]. It is very simple, adds two assumptions to the basic random graph model. One is growth. The vertices and edges are added to the network continuously, instead of allowing every vertex to connect every other in the very beginning. Two is the preferential attachment. When a new vertex connects to some older ones, it chooses based on the (actual) number of connections a vertex has. The higher the degree, the higher the connection probability and the relationship is linear: if the degree is twice as big, then the connection probability is twice as big too.

The model nicely explains the scale-free (in-)degree distribution of the networks, observed widely in very different networks.

See the [Buchanan, 2003, Barabási, 2004, Watts, 2003a,b] popular books and [Newman, 2003a, Albert and Barabási, 2002, Dorogovtsev and Mendes, 2003, Boccaletti et al., 2006] for reviews on network science. [Newman et al., 2006] is an excellent collection of papers.

2

Trait-based networks: a direct problem

At the time we started working with network science as a tool, probably the most obvious feature of real networks that was missing from most of the models studied by mathematicians and physicists were characteristics of individual nodes which influence the connection probability. Thus, if the nodes represent individual persons, it is obvious that in many circumstances two people are more likely to become connected in some form of relationship because of the nature of their individual characteristics. The model we discuss here was motivated by the need to incorporate this idea. A similar idea was used in a preferential attachment model by Bianconi and Barabási [2001] who assigned to each new node a fitness parameter. In their model a larger fitness parameter may overcompensate the smaller probability of attachment.

First we propose a simple model of growing networks whose statistical properties are in some cases identical to a more complicated model containing nodes with distinct characteristics. We will calculate the degree distribution of the growing network, the distribution of cluster sizes and the emergence of a giant cluster. We will also show how the number of attempted connections made when a new node is added determines the position and type of the phase transition as well as the cluster size distribution.

Then we study the model with vertex traits, calculate the degree distribution of this version as well and by numerical simulations show that the traits influence the position of the phase transition where the giant component emerges. We show that the diversity of the trait preferences always favors the formation of the giant component.

2.1 The full, trait-based k-out model

We consider a social network model where each node has individual characteristics or traits. Each node that is added to the network is assigned a permanent set of random traits which could be coded as an ordered binary string or vector of length L. When a node is added it chooses randomly $k \in \mathbb{N}$ possible partners from the already existing nodes, or if there are less then $k+1$ (because the simulation has not yet reached time step $k+2$) it chooses all the existing nodes as possible partners. A trait distance between the new node and one of its possible partners is calculated based on their trait vectors $(\vec{t_1}, \vec{t_2})$ using a distance measure, $D(\vec{t_1}, \vec{t_2})$, such as the Hamming distance. Then a connection is formed between the two nodes with a probability determined from a given probability distribution over the distance function $p(D)$. Different functions, $p(D)$, correspond to different sociopsychological situations. Thus, if we wish to model the case where people are more likely to link together if they have similar traits, then $p(D)$ would be a monotonically decreasing function of D. For this case, the simplest $p(D)$ would be to form a link if D is below some threshold. This procedure is repeated for each possible partner of the new node. Thus, each new node can have initially up to k links with the other existing nodes. Existing nodes can have more than k links as more nodes are added to the network and link up with the existing nodes. There are no multiple links between pairs of nodes.

2.2 The average case

If the trait vectors are drawn uniformly, and the vertices to link to are also chosen randomly, many properties of the network simply depend on the probability δ, that two chosen nodes will link together:

$$\delta = \sum_D p[D(\vec{t_1}, \vec{t_2})] \; r[D(\vec{t_1}, \vec{t_2})] \tag{2.1}$$

where $r(D)$ is the probability of the distance D between two nodes, and the sum is over all possible distance values. Thus, the model is reduced to the following procedure. At each time step we add a node to the network, and attempt to link with k existing nodes which are chosen at random. An actual connection is made with a probability δ. The asymptotic behavior of the network in the limit of large time t, does not depend on the initial condition of starting with a single isolated vertex.

It is clear that some properties of the network depend on the detailed form of $p(D)$ and the nature of the trait vectors. Examples of such properties include the distribution of traits in different parts of the network and the correlation of traits with distance in the network. For example, one can imagine a very simple network of nodes representing men and women. In one network the probability of forming a link is independent of sex, and in the other persons prefer to link up with members of the opposite sex. Clearly, these rules lead to a bipartite network, which never appears if we ignore the traits and calculate only with the average connection probability, δ. We still examine the average model first, as we are interested in if and how (some) structural properties change if we ignore the traits.

2.2.1 The degree distribution

Firstly, we will determine the degree distribution of the network, i.e. the percentage of nodes with m edges. We will use a master-equation approach. Denote by $d_m(t)$ the expected number of nodes with degree m at time t. The number of isolated nodes, $d_0(t)$, will increase by $(1-\delta)^k$, which is the probability of the addition node not connecting to any existing node, and decrease on average by $k\delta d_0(t)/t$:

$$d_0(t+1) = d_0(t) + (1-\delta)^k - k\delta\frac{d_0(t)}{t}. \tag{2.2}$$

The formula for the expected number of nodes of degree $m > 0$ is a bit more complicated. For $1 \leq m \leq k$ there are two ways to increase d_m: either selecting degree $m-1$ nodes for connection with the new node or the new node having exactly m edges. For $m > k$, the new node cannot contribute to d_m. The decrease will be proportional to the probability of choosing a degree m node for attachment.

$$\begin{aligned} d_m(t+1) =& d_m(t) + k\delta\frac{d_{m-1}(t)}{t} + \\ &+ \binom{k}{m}\delta^m(1-\delta)^{k-m} - k\delta\frac{d_m(t)}{t} \qquad \text{if } 1 \leq m \leq k, \qquad (2.3)\\ d_m(t+1) =& d_m(t) + k\delta\frac{d_{m-1}(t)}{t} - k\delta\frac{d_m(t)}{t} \qquad \text{if } m > k. \qquad (2.4)\end{aligned}$$

These equations are correct as $t \to \infty$, and numerical simulations show that $d_m(t) \propto p_m t$. Substituting this form into the equations for $d_m(t)$ we obtain

$$p_m = \delta^m \sum_{j=0}^{m} \binom{k}{j} \frac{(1-\delta)^{k-j}}{(1+k\delta)} \left(\frac{k}{1+k\delta}\right)^{m-j} \qquad \text{if } m \leq k, \tag{2.5}$$

$$p_m = p_k \left(\frac{k\delta}{1+k\delta}\right)^{m-k} \qquad \text{if } m > k. \tag{2.6}$$

The degree distribution p_m decays exponentially.

2.2.2 Critical behavior

2.2.2.1 Cluster size distribution

If a network model is capable of generating unconnected networks, then it is often an important question what is the component size distribution and whether there is a phase transition between a collection of finite size clusters and the appearance of a giant cluster much larger than the rest. The transition is similar to that in percolation, with our parameter δ playing the role of the site occupation probability in a percolation model. The key difference between our model and percolation models is that our nodes do not sit on a lattice structure, and there are thus no geometric constraints. The definition of a giant cluster in our model is somewhat different than a spanning cluster in percolation models. Nevertheless, some of the behavior is similar.

Our model is similar to one by Callaway et al. [2001] where an infinite order phase transition was found. In that model after a node was added to the network, two nodes were picked at random and connected with probability δ. Our model is more general in that we consider the effect of making more than one link at any given time. Also, in our model the new links are between the added node and existing nodes, whereas in the model by Callaway *et al* the new links are between any two nodes in the network.

To determine the cluster distribution we use a procedure similar to the one we used to calculate the degree distribution. The cluster number $N_j(t)$ denotes the expected number of clusters of size j. On average, at each time step, $(1-\delta)^k$ isolated nodes arrive at the network and $k\delta N_1(t)/t$ nodes will be chosen for attachment reducing N_1. Thus, N_1 is described by

$$N_1(t+1) = N_1(t) + (1-\delta)^k - k\delta\frac{N_1(t)}{t}. \tag{2.7}$$

For $j > 1$ new clusters of size j come from connecting the new node to a cluster of size $j-1$ or if $k > 1$ using the new node to make connections between smaller clusters whose sizes, together with the new vertex, add up to j. Reducing N_j will be $jk\delta N_j(t)/t$ nodes from clusters of size j connecting to the new node. Thus, we have

$$N_2(t+1) = N_2(t) + \binom{k}{1}\delta(1-\delta)^{k-1}\frac{N_1(t)}{t} - k\delta\frac{2N_2(t)}{t} \tag{2.8}$$

$$\vdots$$

$$N_j(t+1) = N_j(t) + \sum_{r=1}^{\min(k,j-1)}\binom{k}{r}\delta^r(1-\delta)^{k-r}S(j,r) - k\delta\frac{jN_j(t)}{t}. \tag{2.9}$$

with

$$S(j,r) = \sum_{\substack{z_1+\ldots+z_r=j-1\\ z_i\geq 1,\ i\leq r}} \frac{z_1N_{z_1}(t)}{t}\cdots\frac{(j-1-\sum_{i=1}^{r-1}z_i)N_{j-1-\sum_{i=1}^{r-1}z_i}(t)}{t} \tag{2.10}$$

The first sum in Eq. (2.9) determines the number of sums in the next term, $S(j,r)$. Each of these sums represent a cluster that is melted into the j sized cluster. These equations are valid for $t\to\infty$, where the probability of closed loops tends to zero. The giant cluster, if there exists one, is an exception in which connection of nodes in loops is not negligible. Thus, Eq. (2.9) holds only for the finite sized clusters in the network. This property lets us determine a generating function which we can use to find the size of the giant cluster. Our simulations show that solutions of Eqs. (2.7), (2.8) and (2.9) are of the steady state form $N_j(t) = a_jt$. Using this form in Eqs. (2.7), (2.8) and (2.9), we find

$$a_1 = \frac{(1-\delta)^k}{1+k\delta} \qquad a_2 = \frac{\binom{k}{1}\delta(1-\delta)^{k-1}a_1}{(1+2k\delta)} \tag{2.11}$$

$$a_j = \frac{1}{1+jk\delta}\sum_{r=1}^{\min(k,j-1)}\binom{k}{r}\delta^r(1-\delta)^{k-r}\cdot \cdot\sum_{\substack{z_1+\ldots+z_r=j-1\\ z_i\geq 1,\ i\leq r}}\big(j-1-\sum_{i=1}^{r-1}z_i\big)a_{j-1-\sum_{i=1}^{r-1}z_i}\prod_{l=1}^{r-1}z_la_{z_l}. \tag{2.12}$$

Generally we cannot obtain a simpler equation for the cluster size distribution a_j, except for $k=1$. Substituting $k=1$ into the Eqs. (2.11) and (2.12) we

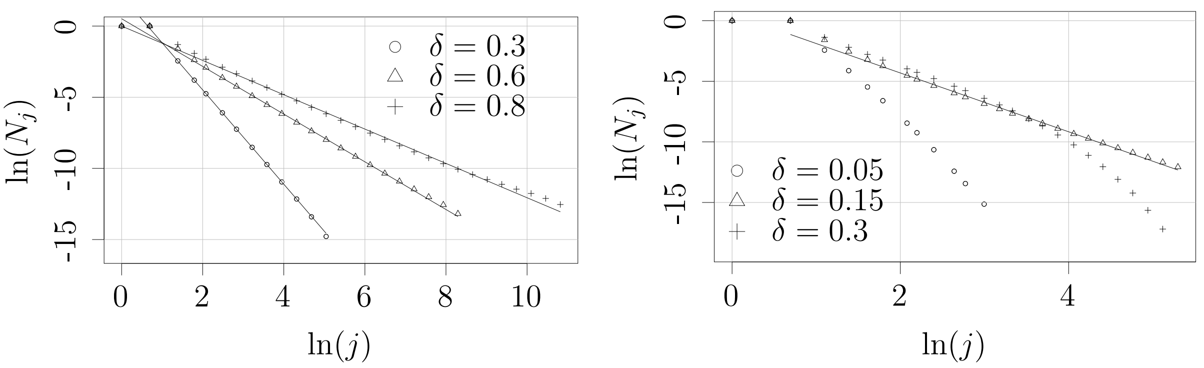


Figure 2.1: Cumulative cluster size distribution for different δ-s and $k = 1$ (left), $k = 2$ (right). The cumulative cluster size distribution has less noise and it is a power-law if and only if the non-cumulative cluster size distribution is a power-law as well. Solid, dashed and dotted lines are obtained from a least squares fit indicating the power-law behavior of the distributions. Simulation data were obtained by averaging over 500 runs of 10^6 time-steps and are shown on a log-log plot. Note that in the right figure simulations for $\delta = 0.05$ and $\delta = 0.3$ distributions do not follow a power-law. In Section 2.2.2.2 it is shown that there is a phase transition near $\delta = 0.146$.

obtain after some algebra the general result

$$a_j =(1-\delta)\delta^{j-1}(j-1)!\prod_{m=1}^{j}\frac{1}{1+m\delta}, \tag{2.13}$$

which can be written in the form:

$$a_j =\frac{(1-\delta)\Gamma(1/\delta)}{\delta^2}\frac{\Gamma(j)}{\Gamma(j+1+1/\delta)}, \tag{2.14}$$

where $\Gamma(x)$ denotes the gamma-function. Eq. (2.14) shows that the cluster size distribution for $k = 1$ always follows a power-law distribution. This result is confirmed by simulations shown in the left graph of Fig. 2.1. Distributions of cluster sizes for $k = 2$ (right graph of Fig. 2.1), in contrast to $k = 1$ show power-law behavior only near the phase transition.

2.2.2.2 Position of the phase transition

Fig. 2.2 shows the simulation results for S, the ratio of the average size of the largest cluster to the total number of nodes versus the connection probability δ. The figure suggests that there is a smooth transition in the appearance of S at a specific value of δ between $\delta = 0$ and $\delta = 0.2$, which depends on the parameter k. To predict the position of a possible phase transition δ_c [Callaway et al., 2001], we will use a generating function for the cluster size distribution [Wilf, 1994]. To derive the generating function we use the iterative Eqs. (2.11), and (2.12). The generating function will be of the form:

$$g(x) = \sum_{j=1}^{\infty} b_j x^j, \tag{2.15}$$

where

$$b_j = j a_j, \tag{2.16}$$

is the probability that a randomly chosen node is from a cluster of size j. Multiplying both sizes of Eqs. (2.11) and (2.12) by jx^j, and summing over j we derive a differential equation for $g(x)$

$$g = -k\delta g\prime + x(1-\delta)^k + \sum_{i=1}^{k} \binom{k}{i} \delta^i (1-\delta)^{k-i} (x^2 g\prime g^{i-1} i + x g^i). \tag{2.17}$$

Rearranging for $g\prime$ we obtain

$$g\prime = \frac{(1-\delta)^k - g/x + \sum_{i=1}^{k} \binom{k}{i} \delta^i (1-\delta)^{k-i} g^i}{k\delta - x \sum_{i=1}^{k} \binom{k}{i} \delta^i (1-\delta)^{k-i} g^{i-1} i}, a \tag{2.18}$$

which can be further simplified to

$$g\prime = \frac{-g/x + (1+(g-1)\delta)^k}{k\delta - xk\delta(1+(g-1)\delta)^{k-1}}. \tag{2.19}$$

The generating function for the finite size clusters is exactly one at $x = 1$ when there is no giant cluster in the network and $g(1) < 1$ otherwise. Hence

$$S = 1 - g(1). \tag{2.20}$$

Without an analytic solution for Eq. (2.19), we calculate S numerically by integrating Eq. (2.19) with the initial condition $(x, g(x)) = (x_0, x_0(1-\delta)^k/(1+k\delta))$ where x_0 is small. This is equivalent to starting with a cluster of only one node. In Fig. 2.2 there are results from direct simulations of the model (symbols) and solid lines from the integration of the generating function. The agreement is good which verifies the approximations.

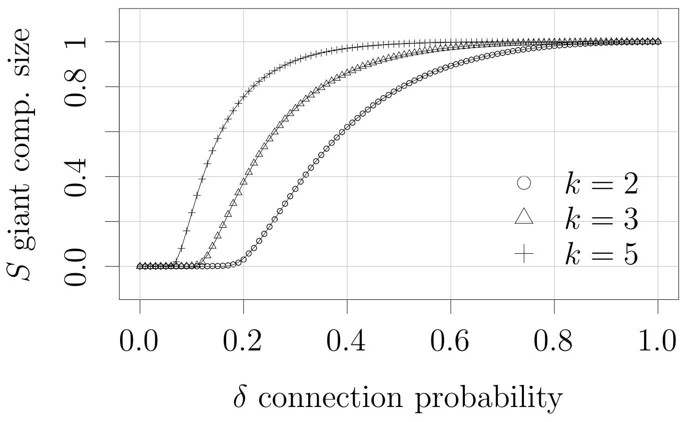


Figure 2.2: Giant cluster size S as a function of δ and k. Symbols are from simulations of the growing network for 10^6 time steps averaged over 30 runs. Lines are from the analytical calculations.

To discuss the phase transition location we first consider the cases $k > 1$. Consider the expected value that a randomly chosen node belongs to a finite size cluster. We can determine this quantity in terms of the generating function $g(x)$

$$\langle s \rangle = \frac{g\prime(1)}{g(1)}. \tag{2.21}$$

For those values of δ where no giant cluster exists, $\delta < \delta_c$, $g(1) = 1$, and both the numerator and denominator of Eq. (2.19) goes to zero as $x \to 1$. Using L'Hopital's rule we derive a quadratic equation for $g\prime(1)$. The solution of this equation is

$$g\prime(1) = \frac{1 - 2k\delta \pm \sqrt{(2k\delta - 1)^2 - 4k(k-1)\delta^2}}{2k(k-1)\delta^2}, \tag{2.22}$$

for $g(1) = 1$. Because as $\delta \to 0$ all clusters will have size 1, one can show that the correct solution of Eq. (2.22) is the one with the negative sign. In addition from Eq. (2.22) we can find the location of the phase transition. It is the value of δ where the solution of Eq. (2.22) becomes complex:

$$\delta_c = \frac{1 - \sqrt{1 - 1/k}}{2}. \tag{2.23}$$

In the region where there is a giant cluster $\delta > \delta_c$, Eq. (2.19) becomes as $x \to 1$,

$$g\prime = \frac{-g + (1 + (g-1)\delta)^k}{k\delta - k\delta(1 + (g-1)\delta)^{k-1}}, \tag{2.24}$$

which is still not solvable analytically. Making the approximation $(1 \pm a)^k \approx 1 \pm ka$ when $a \ll 1$, we can simplify Eq. (2.24) close to δ_c:

$$g\prime(1) \approx \frac{k\delta - 1}{k\delta^2(1-k)}, \tag{2.25}$$

where $g(1) < 1$, $\delta > \delta_c$, and $(g(1) - 1)\delta \ll 1$. In Fig. 2.3 we show the simulation results and the above derived theoretical functions for $g\prime(1)$. We can see that for $\delta < \delta_c$, where we have an explicit expression for $g\prime(1)$ in terms of the parameters k and δ the fit is very good. For $\delta > \delta_c$ the fit is good close to the phase transition point, where the approximation $(g - 1)\delta \ll 1$ holds. Although below δ_c the description of $g\prime(1)$ is very good, it seems that the location of the phase transition and the value of the function $g\prime(1)$ above δ_c is somewhat different than the data. Also if we carefully check Fig. 2.3 at the jumps, we find that the larger the jump the less accurate the theory seems to be. This can be explained as follows. At the critical point the average size of finite clusters jumps, hence much larger clusters appear in the network. As we can only simulate for a finite time large (but not the giant) clusters are underrepresented. The weights of them computed from the simulation data are less then they would be in an infinitely long simulation. Away from the transition regime fewer finite size clusters remain beside the giant cluster in the network, and thus the distribution can be specified better.

Although the formalism using the generating function can be done for $k = 1$, the meaning of a giant cluster is problematic. In Section 2.2.2.1 we showed that the size-distribution of clusters for $k = 1$ always follows a power-law which means there is no obvious border between the 'giant' cluster and smaller clusters. There is not a sharp break between the largest and the next largest cluster. The physical reason for this is that clusters grow only by the addition of newly added nodes. This is different than the case for $k > 1$ and in percolation models where clusters can also grow by a link combining two clusters. In this sense no giant cluster appears in the network except for $\delta = 1$. Eq. (2.19) becomes

$$g\prime(x) = \frac{(1-\delta) - g/x + \delta g}{\delta(1-x)}, \tag{2.26}$$

which becomes $\frac{0}{0}$ in the limit $x \to 1$ with $g(1) = 1$. Applying L'Hopital's rule yields

$$g\prime(1) = \frac{1}{1-2\delta}. \tag{2.27}$$

At $\delta = \frac{1}{2}$, $g\prime(1) \to \infty$, which means the average size of finite clusters approaches infinity. From the definition of $g(x)$ in Eq. (2.15) and the power-law

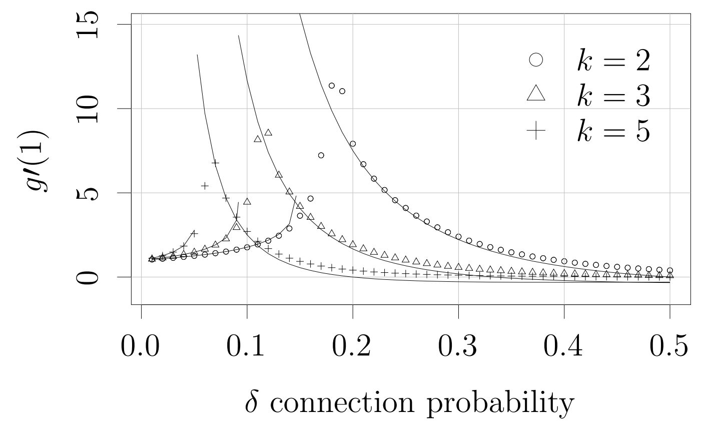


Figure 2.3: Discontinuity in $g\prime(1)$ for different values of k. Solid lines are theoretical, and symbols are results from the simulations of growing networks for 10^6 time steps, averaged over 30 runs.

cluster size distribution for a_j, it follows that $g(1) = 1$ for any $\delta \neq 1$. To see that $g\prime(1) \to \infty$ as $x \to 1$ for $\delta > \frac{1}{2}$, we consider the sum form of the generating function in Eq. (2.15). For large j, $a_j \approx 1/(j^{1+1/\delta})$ Eq. (2.14), and $g\prime = \sum_{j=1}^{\infty} j^2 a_j x^{j-1}$, which can not be summed for $\delta \geq \frac{1}{2}$.

When $\delta < \frac{1}{2}$, the probability of a new node not joining a cluster is higher then joining, and thus the weight of small clusters is higher than that of larger clusters, and hence the average size remains finite. As $\delta \to \frac{1}{2}$, the probability of forming clusters increases and so do the weight of large clusters.

2.2.2.3 Infinite-order transition

To show the nature of our phase transitions [Callaway et al., 2001], we numerically integrated Eq. (2.19) for different values of k near the corresponding critical δ_c. In Fig. 2.4 the linear parts of the log(-log(S)) plots suggest that

$$S(\delta) \propto e^{\alpha(\delta-\delta_c)^\beta} \qquad \text{as } \delta \to \delta_c, \tag{2.28}$$

and because all derivatives of S vanish at δ_c, the transition is of infinite order.

Table 2.1 contains the parameters of the fitted straight lines in Fig. 2.4. As the calculations were done close to the numerical limit and referring to the similar results by Callaway et al. [2001] we conjecture that $\beta = 1/2$ for all k. This result suggests that the mechanism of the transition is common and the number of possible partners for each node to link to determines the speed of emergence of the giant cluster S. These results are in accord with

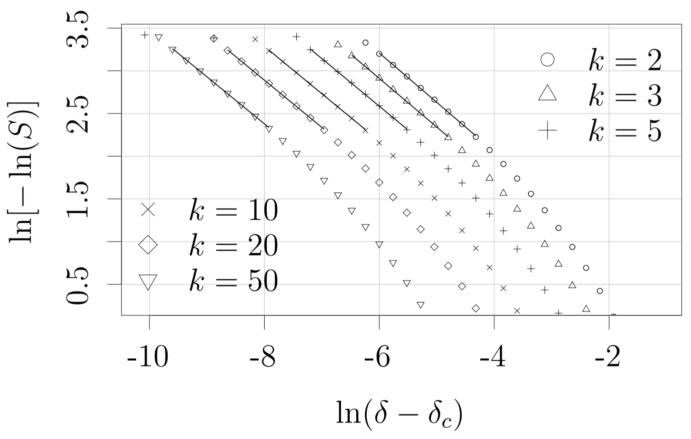


Figure 2.4: Numerical calculation of the giant cluster size close to but above the phase transition. Least-squares fitted solid straight lines suggest $S(\delta) \propto e^{\alpha(\delta-\delta_c)^\beta}$. The flat ends of the curves on the top appear due to the limit of the accuracy of numerical integration.

k	2	3	5	10	20	50
α	-0.256	-0.504	-0.758	-1.149	-1.530	-2.023
β	-0.577	-0.569	-0.557	-0.554	-0.552	-0.550

Table 2.1: The parameter values (α and β) of the fitted lines in Fig. 2.4. Taking into account that we were at the border of the maximal numerical accuracy and that the fit is short we presume $\beta = -\frac{1}{2}$.

Eq. (2.25), the average cluster size decrease is approximately independent of k, but the size of the jump and the rate of decrease is driven by k.

2.2.3 Discussion of the average case

The presented model was intended to gain insight into the evolution of various social networks by considering mechanisms that account for heterogeneity in the population of participating entities. To analyze the statistical properties of the generated network we simplified the model. We found that the structure of the network dramatically changes when the number of possible links to a newly added node increases from $k = 1$ to $k = 2$. With $k = 1$ the network does not form a giant cluster but the average cluster size goes to infinity (at $\delta = \frac{1}{2}$) in contrast to $k \geq 2$, where the giant cluster appears in an infinite-order phase transition and the average cluster size jumps discontinuously but remains finite. The size of the jump corresponds to how slowly the giant cluster overcomes the other competitive large clusters. However, there is no transition for $k = 1$, where none of the clusters can absorb other clusters. The distribution of the size of finite clusters always follows an exponential distribution, both below and above the critical point for $k > 1$, while the model studied in [Dorogovtsev et al., 2001a, Callaway et al., 2001] is in a critical state below and at the critical point and exhibits an exponential distribution of cluster size above the transition as in a Berezinskii-Kosterlitz-Thouless phase transition. Thus, even though there are disconnected clusters as in our model, there are significant differences in the behavior of the cluster size distribution.

Our model is similar to a previous model of Callaway et al. [2001], but there are essential differences in several points due to nature of the growth algorithm: in the model of Callaway *et al.* network growth and connection formation are independent while in our model only newly added nodes form connections. Also, in our model multiple connections might be formed in one time step depending on parameters k and δ. This difference is well reflected in the generating function derived for the two models.

2.3 The effect of the traits

2.3.1 The matrix-based description

We now turn back to the full trait-based model to see the effect of the traits compared to the averaged, δ-based mode. Let us now assume that there are n different possible trait vectors, and n is finite. The model can be then described by using a vector (r) of length n giving the distribution of the different trait vectors and an $n \times n$ matrix (Δ) giving the connection probabilities of the trait vectors, k is the usual parameter.

We enumerate the possible trait vectors, a vertex is called a *type i* vertex if its trait vector is the i-th in the enumeration. If the r_i $(0 \leq i \leq n)$ (column) vector and the Δ_{ij} $(0 \leq i, j \leq n)$ matrix are given then the δ-based counterpart of a trait-based model can be obtained as

$$\delta = r^T \cdot \Delta \cdot r, \tag{2.29}$$

where the T superscript denotes the transpose.

2.3.2 Degree distribution

Let us know calculate the degree-distribution of the trait-based model. By $d_k^{(i)}(t)$ we denote the expected value of the degree of a vertex of type i. The following equations are valid if $m > k$:

$$d_0^{(i)}(t+1) = d_0^{(i)}(t) + r_i[\sum_{j=1}^{n}(1-\Delta_{ij})r_j]^k - k[\sum_{j=1}^{n}\Delta_{ij}r_j]\frac{d_0^{(i)}(t)}{t}. \tag{2.30}$$

$$d_m^{(i)}(t+1) = d_m^{(i)}(t) + k[\sum_{j=1}^{n}\Delta_{ij}r_j]\frac{d_{m-1}^{(i)}(t)}{t} - k[\sum_{j=1}^{n}\Delta_{ij}r_j]\frac{d_m^{(i)}(t)}{t}. \tag{2.31}$$

The corresponding equations for $1 \le m \le k$ are more difficult, since combinatoric terms appear. They are not important however as we're interested in the behavior of the tail of the degree distribution. Simulations show that the degree distribution is stationary, $d_n^{(i)} = p_n^{(i)}t$, so we have

$$p_m^{(i)} = p_k^{(i)}(\frac{ka_i}{1+ka_i})^{n-k}, \qquad a_i = \sum_{j=1}^{n}\Delta_{ij}r_j. \tag{2.32}$$

This means that the degree distribution is the sum of exponential distributions, and the tail follows the slowest decaying term, the vertex type with the highest a_i "attractiveness" dominates.

Thus, although the quantitative behavior is different in the full model than the reduced model, the qualitative behavior is the same and it is a basic characteristic of the models based on uniformly random choice.

2.3.3 The giant cluster

Unfortunately it is in practice too difficult to carry out the calculations based on master equations and generating functions to exactly determine the position of the phase transition in the full trait-based model.

While the trait-based case might have the same kind of infinite order phase transition, the position of the phase transition is surely affected by fine structure of the trait distribution vector and the trait preference matrix. We demonstrate this with a simple model. Let us assume that we have two types of vertices and the trait distribution vector and the preference matrix are given as

$$r = \begin{bmatrix} x \\ 1-x \end{bmatrix}, \qquad \Delta = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \tag{2.33}$$

with the single parameter x. Then, according to Eq. (2.29) x can be calculated from the δ average connection probability as $x = 1 - \sqrt{1-\delta}$.

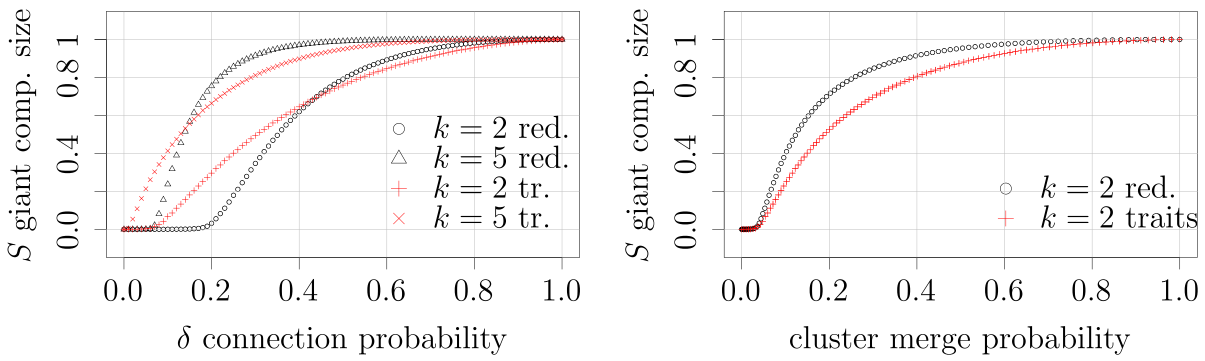


Figure 2.5: Left plot: comparing the giant cluster size S in the reduced, averaged and the full, trait-based models. Right plot: the giant cluster size in function of μ. The μ cluster merge probability is a much better predictor for the position of the phase transition than δ. Both plots show the average of 30 simulations, each simulation had 10^6 vertices.

We simulated the averaged model and the trait based model with the same average connection probability and by varying δ compared the phase transition positions in the two models. The results are shown in Fig. 2.5.

Clearly, the structure of our Δ matrix favors the generation of large clusters, since there is a vertex type, which always connects to the chosen vertices and this vertex type tends to dominate in the larger clusters. It is a natural question whether the vertex correlation introduced by the non-uniform attachment probabilities always favors the formation of the giant component, or the phase transition position can be also “shifted” to the right. This actually includes two questions:

1. Is the transition threshold always smaller in the trait-based model?
2. If δ is fixed then is the giant cluster always bigger in the correlated network?

The second question is easily answered by Fig. 2.5, which shows both possibilities. Let us deal with the first question then.

In the following we assume that $k = 2$. From Fig. 2.5 we know that the δ parameter does not accurately predict the position of the phase transition if degree correlations are present. A better predictor is the probability that a new vertex connects two clusters, we denote this by μ. Clearly, in the reduced model $\mu = \delta^2$. In the trait-based model we can estimate μ by defining the attractiveness of the vertices:

$$a_i = \sum_{j=1}^{n} r_j \Delta_{ij}, \tag{2.34}$$

this is the probability that a vertex of type i connects to a randomly chosen vertex. Then we have

$$\delta = \sum_{i=1}^{n} r_i a_i, \qquad \mu \approx \sum_{i=1}^{n} r_i a_i^2. \tag{2.35}$$

See the second figure in Fig. 2.5 for experimental evidence in favor of using μ for predicting the phase transition position.

Let us now fix the $r > 0$ vector (if $r_i = 0$ then this vertex type can be simply omitted) and discuss the behavior of μ in the function of the a_i $(1 \leq i \leq n)$ values at a fix δ average connection probability. We want to solve the minimization/maximization problems

$$\max_a \sum_{i=1}^{n} r_i a_i^2, \qquad \min_a \sum_{i=1}^{n} r_i a_i^2, \qquad \sum_{i=1}^{n} r_i a_i = \delta, \tag{2.36}$$

$0 \leq a_i \leq 1$ and $1 \leq i \leq n$. This is actually quite easy with the Lagrange multiplicator method. The Lagrange function is

$$\mathcal{L}(a_1, \ldots, a_n, \lambda) = \sum_{i=1}^{n} r_i a_i^2 + \lambda[\sum_{i=1}^{n} r_i a_i - \delta], \tag{2.37}$$

its partial derivatives are

$$2 r_i a_i + \lambda r_i = 0, \qquad 1 \leq i \leq n, \qquad \sum_{i=1}^{n} r_i a_i - \delta = 0. \tag{2.38}$$

If $r_i > 0$ for all $1 \leq i \leq n$ then the only solution of this is $a_i = \delta$ for all $1 \leq i \leq n$. With some simulations it is easy to see that this is a minimum point, μ is the smallest possible if all the vertices have the same attractiveness, which naturally happens in the δ-based model.

We have shown that any kind of correlation introduced to the trait-based model (compared to the averaged, reduced case) increases the probability that a new vertex connects two clusters, and thus reduces the threshold of the phase transition to a giant cluster. It is clear from the right plot of Fig. 2.5 that our estimation based on the μ join probability is not exact, it does not take into account the distribution of the traits within the clusters. We still believe that our finding is an important result.

In parallel to our work, and also later, many researchers studied trait-based network models, although in most cases static networks. In particular, Bollobás et al. [2005] proved our result about the infinite order phase transition rigolously, see also [Bollobás and Riordan, 2005]. Other such works include [Söderberg, 2002, 2003a,b,c, Boguna and Pastor-Satorras, 2003, Bollobás et al., 2007].

3

The inverse problem: reverse engineering the evolution of networks

THIS CHAPTER CONTAINS the main part of my dissertation. In Section 3.1 we first introduce a simple model framework for *kernel-based* citation networks and generalize that to wider classes of networks. Then in Sections 3.2 and 3.3 give two solutions to the inverse problem of best describing the dynamics of a given network in the kernel-based framework.

3.1 The model framework

3.1.1 Introductory remarks

The kernel-based framework use attributed simple graphs. Recall that a graph is simple if it does not contain multiple and loop edges and in an attributed graph vertices are labeled with strings from a given alphabet.

The model framework is a discrete time, discrete space, stochastic system. The discrete space seems obvious, finite graphs are discrete objects after all. If a graph possesses continuous vertex attributes then we require the discretization of these. Discrete time means that we will define time steps which serve as snapshots of the dynamic graph. It is analogous to a movie composed of still images: we create snapshots of the dynamic graph and describe the changes between the consecutive still images. All changes in the network, addition and deletion of edges and vertices happen *between* the time steps. Finally, stochastic means that the description does not give exactly how a given graph changes in time, it gives only the probabilities of all possible changes. As the reader familiar with Markov chains surely noted already, the model framework is basically a Markov chain where the state of the system is an attributed graph.

3.1.2 Citation networks

We first define the model framework for citation networks. Citation networks form a subset of all directed evolving networks, they are special in two respects, see also Fig. 3.1 for an example citation network.

1. Edges and vertices are never deleted from the network. This allows us to define two functions, t^V and t^E which give the last time step (i.e. the snapshot) when a given vertex or edge is *not* present in the network. If vertex v appears first in time step s, then $t^V(v) = s - 1$.

   The reason for this strange notation is that whenever we want to measure the probability that a given vertex or edge is added we do this based on the state of the network right *before* adding the edge.

   Obviously, the t^V and t^E functions must be compatible, which means that the end points of an edge must be added before the edge itself, if $e = v \rightarrow w$ then $t^V(v) \leq t^E(e)$ and $t^V(w) \leq t^E(e)$.

   In the following, we omit the E and V indices if it is clear whether the argument is an edge or a vertex.

   We denote the graph at time step t by $\mathbb{G}(t)$. $V(\mathbb{G}(t))$ is the set of all vertices in the network at time step t.

2. All outgoing edges of a vertex are added right after the vertex itself. In the snapshot analogy this means that if there are two vertices, v and w, and v appears before w on the snapshots, $t^V(v) \leq t^V(w)$, then all the outgoing edges of v must also appear before w, $t^E(e) \leq t^V(w)$ for all outgoing edges e of v.

From now on, we will assume that a single vertex is added to the citation network in each time step.

We are ready to define the first kernel function, which tells the probability that a given edge cites a given vertex: $A(\cdot)$. Note that this single kernel function does *not* describe whether or when a given vertex is added to the

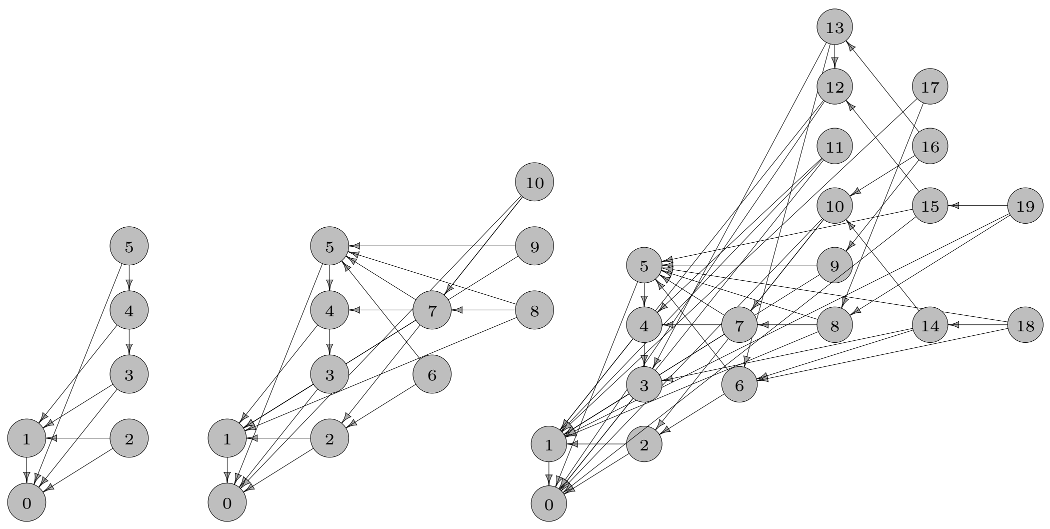


Figure 3.1: Some snapshots for a citation network. Since the new vertices are always added to the right and to the top of old vertices, all edges go to the left or downwards. Here we show three snapshots, vertices 6-10 are added *between* the first and the second and 11-19 *between* the second and the third. Notice that all outgoing edges of a vertex are added with the vertex itself.

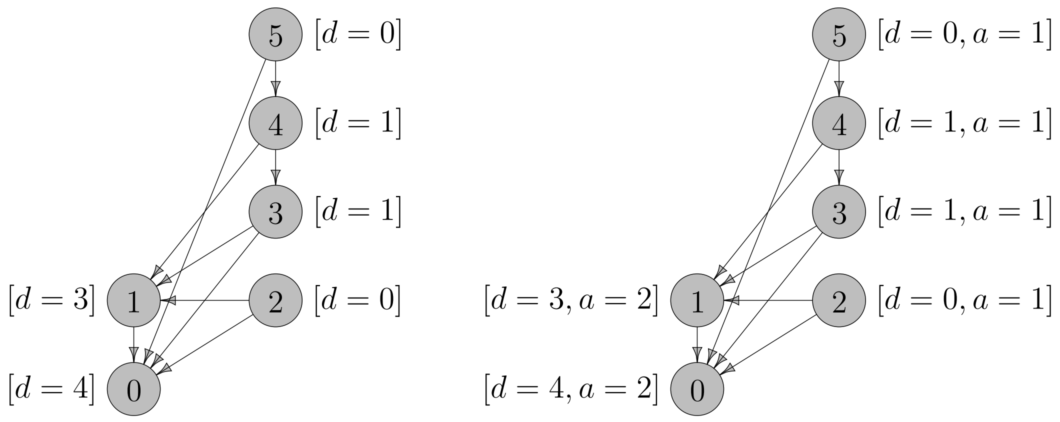


Figure 3.2: Examples for property vectors in citation networks. In the first network the vertices have a single property, their in-degree. In the second network a second property is added to the property vector, the "age" of the vertices. The age of a vertex is defined as the number of time steps passed since the vertex was added to the network.

network, nor the number of citations it makes. But if we know that a citation is made by a certain vertex then it gives the probability that this single citation goes to a given vertex.

The kernel function depends on the properties of the potentially cited vertices. These properties can be either structural: the degree or some other centrality score of the vertices, their local transitivity, etc.; or non-structural: the age of the actors in a social network or the topic of the web-pages on the WWW, etc. The properties might as well be time-dependent. See Fig. 3.2 for property vector examples. Note that in this description the citing vertex is passive, the citation probability only depends on the properties of the potentially cited vertices. We will fix this shortcoming later.

Sometimes it is easier to talk about vertex types rather than property vectors. If there are n possible different property vectors, then we just say that there are n vertex types, denoted by natural numbers up to n. Even if the number of possible property vectors is infinite (e.g. vertex degree can be arbitrarily large), only a finite subset of them appears during the evolution of a given network. If we assign a property to the vertices which may take continuous values, then for solving the inverse problem (see later) these values must be discretized first, for practical reasons. In the following, the term i-vertex (e.g. 1-vertex) means a vertex having property vector of type i (type 1 for a 1-vertex).

Now, we define the probability that some vertex with a given property vector x (vertex type x if you like) is cited by an edge. This is the heart of the framework. $c(e,x)$ are indicator random variables bound to a fixed network. Indicator random variables are binary: $c(e,x)$ is 1 if and only if edge e cites a vertex of type x, otherwise it is 0. The model framework defines the probability that $c(e,x)$ is one:

$$P[c(e,x)=1] = \frac{A(x)N(t(e),x)}{\sum_{j\in V(\mathbb{G}(t(e)))} A(x_j(t(e)))}. \tag{3.1}$$

Here $A(x)$ is the attractiveness (kernel function value) of a single x-vertex, $N(t(e),x)$ is the number of x-vertices in time step $t(e)$, and $x_j(t(e))$ is the property vector of vertex j in time step $t(e)$. $\mathbb{G}(t(e))$ is the graph at time step $t(e)$ and $V(\mathbb{G}(t(e)))$ is its set of vertices. From now on, we will often denote the normalization factor by $S(t(e))$:

$$S(t(e)) := \sum_{j\in V(\mathbb{G}(t(e)))} A(x_j(t(e))). \tag{3.2}$$

Please note that an $A(\cdot)$ kernel function is the same in all respects as $cA(\cdot)$, where c is a positive number. This means that the value of $A(\cdot)$ for *one* vertex type can be chosen arbitrarily, we will often use this fact and either choose $A(x)=1$ for some x vertex type or $\sum_i A(i)=1$, where i goes over all vertex types.

Observing a single edge e in a given network is an experiment performed on the network, and the goal of these experiments is to determine $A(\cdot)$.

The inverse problem (Fig. 3.3.) means to construct an $A(\cdot)$ function that *well* describes the evolution of the network. Certainly, this makes sense only if we are able to define what *well* means here. A natural definition would be the probability that the given kernel function generates *exactly* the network $\mathbb{G}$ under study, or more precisely, the probability that the kernel function is able to guess the outcome of all edge experiments:

$$P[A(\cdot) \text{ generates } \mathbb{G}] = \prod_e \frac{A(x_e)}{S(t(e))}. \tag{3.3}$$

Here, x_e is the type (i.e. property vector) of the vertex cited by edge e (at time step $t(e)$). If these probabilities are (relatively) high, then the kernel function fits well the network.

For a large network these probabilities are very small (even if relatively high), we take the logarithm of Eq. 3.3 and translate it by subtracting the logarithmic goodness of the totally random attachment. This way, a kernel

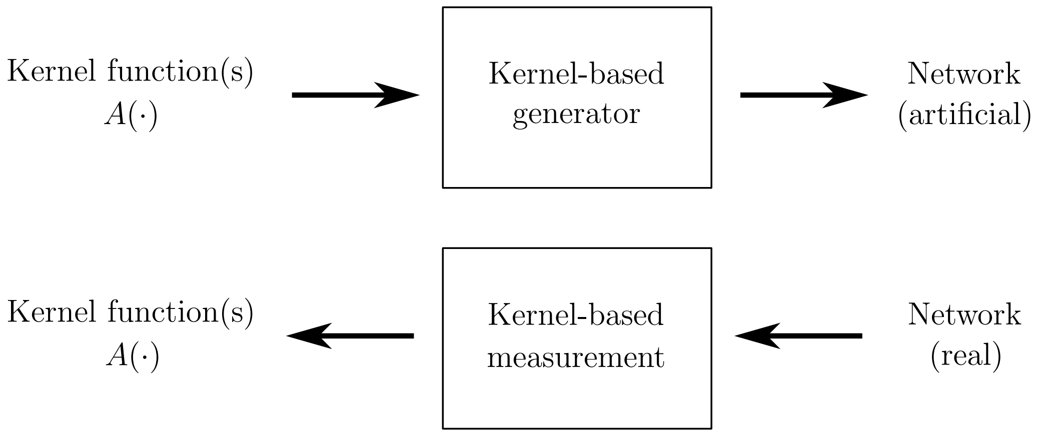


Figure 3.3: The direct and the inverse problems. In the direct problem we have the $A(\cdot)$ kernel function (and maybe other kernel functions for more general networks) and we use a kernel-based network generator algorithm to create an artificial network. The question is what kind of structures we can obtain from a given class of kernel functions.
In the inverse problem we examine a real network, assembled from observations. The kernel-based measurement algorithm takes this as the input and gives a *possible* kernel function as the output. If the measurement algorithm is good then the output kernel function is *good*, i.e. it creates the observed network with high probability.

function with goodness value zero tells absolutely *no information* about the network. We also divide by the number of edges to obtain a value typically between zero and one. Thus, the goodness of $A(\cdot)$ is defined as:

$$\frac{1}{|E|}\left[\sum_e \log\frac{A(x_e)}{S(t(e))} - \sum_e \log\frac{1}{t(e)}\right]. \tag{3.4}$$

Note that the goodness score is not an *absolute* measure, it makes no sense to compare the goodness of kernel functions for different networks. It makes, however, sense to compare the goodness of different kernel functions on the same network. By this way, alternative descriptions of a system can be created and we have a quantitative measure to tell which one is a better. We show example applications for this in Section 4.

Assuming we have a procedure to find the kernel function with the highest goodness on a given set of vertex properties, we can also tell whether it is worth to extend a set of properties with an additional one: if the best kernel function for the larger set is not substantially better than the one for the smaller set, the extension is not needed, as it adds nothing (or almost nothing) to the goodness.

See Sections 3.2 and 3.3 for methods to solve the inverse problem. In the rest of this Section we generalize the framework in three steps: first, we take into account the type of the *citing* vertices in citation networks, secondly, we consider growing networks, i.e. networks without edge and vertex deletion, and thirdly, general networks with arbitrary combinations of edge and vertex additions and deletions.

3.1.3 Type of the citing vertex

The framework introduced in the previous section ignores the type of the citing vertex when defining the probabilities. It is very easy, however, to overcome this. We can simply define different kernel functions for different citing vertex types, and in each time step the kernel function corresponding to the type of the citing vertex is effective.

Note that it is not necessary to define the same vertex types for the citing and the cited vertices, in practice they usually have different types.

Using the types of the citing vertices provides a simple way to create time dependent kernel functions. If we are only interested in the dynamics of a given time period, then we can define two (citing) vertex types, one inside and the other outside the time period in question, and the two kernel functions can be compared. By using a sliding time window for defining the vertex types, we can have a series of smoothly changing – or not changing, if the dynamics is stationary – kernel functions.

3.1.4 Growing networks

This section generalizes the framework defined for citation networks to general growing networks. In these it is still true that edges and vertices are never deleted. It is, however, not necessarily true that the adjacent edges of a newly added vertex are added right after adding the vertex itself. The networks discussed in this section can be either directed or undirected. Also, it is no longer true that a single vertex is added to the network in each time step. Instead, any number of vertices and edges might be added in a time step.

Now, the kernel function $A(\cdot,\cdot)$ depends on the properties of the two potentially connected vertices and it is symmetric for undirected networks. The $c(e,x,y)$ indicator random variable is 1 if and only if edge e connects a x-vertex to a y-vertex. The model framework gives the probability that $c(e,x,y)=1$:

$$P[c(e,x,y)=1] = \frac{A(x,y)N(t(e),x,y)}{S(t(e))}. \tag{3.5}$$

The equation is analogous to the citation network case, $N(t(e),x,y)$ gives the number of *missing* edges between x- and y-vertices, for undirected networks it is symmetric in the second and third variables. The normalization factor in the denominator goes over all pairs of vertices which could be possibly connected. For directed networks it is:

$$S(t(e)) = \sum_{i,j=1}^{n} N(t(e),i,j)A(i,j), \tag{3.6}$$

for undirected ones we need a slight modification to count every possibility only once:

$$S(t(e)) = \sum_{\substack{i,j=1 \\ i\le j}}^{n} N(t(e),i,j)A(i,j). \tag{3.7}$$

The number of vertex types is denoted with n, as usual.

3.1.5 The most general framework

Just like edge addition can be described by defining the addition probabilities based on a simple kernel function, the same can be easily done with edge deletion as well. If we know that some edge is being deleted from the network (we denote the to-be-deleted edge by e), then the probability that an edge connecting an x-vertex to a y-vertex will be deleted is given as

$$P[d(e,x,y)=1]=\frac{D(x,y)N^d(t(e),x,y)}{S^d(t(e))}, \tag{3.8}$$

where $D(x,y)$ is the kernel function governing edge deletion, $N^d(t(e),x,y)$ is the number of edges connecting x- and y-vertices in time step $t(e)$, for undirected graphs it is symmetric in the second and third variables, and the $S^d(t(e))$ normalization factor is defined as

$$S^d(t(e))=\sum_{i,j=1}^{n}N^d(t(e),i,j)D(i,j) \tag{3.9}$$

for directed networks and as

$$S^d(t(e))=\sum_{\substack{i,j=1\\i<j}}^{n}N^d(t(e),i,j)D(i,j) \tag{3.10}$$

for undirected networks.

If in a given network we want to model both the addition and the deletion of the edges then we have two kernel functions, $A(\cdot,\cdot)$ for the additions and $D(\cdot,\cdot)$ for the deletions. Note that these two are formally not connected and this is intentional. In general, we cannot know whether edge additions and edge deletions are related in a given network, thus we *must not* assume anything about their relation. Naturally, after the measurement was done, it may turn out that they are connected in some way. Note that it is not required to model edge addition and deletion based on the same vertex properties.

Even if we have the $A(\cdot,\cdot)$ and $D(\cdot,\cdot)$ kernel functions for a given network, that is still not a complete description in the sense that we don't know how vertices are added and deleted and we also don't model which of these edge/vertex deletion and addition events happen at any given time. So far, we only know that *if* an edge is added then it is added according to $A(\cdot,\cdot)$, and *if* an edge is deleted then it is deleted according to $D(\cdot,\cdot)$.

It is not difficult to create the remaining required kernel functions:

1. $A^v(\cdot)$ describes what kind of vertex is added to the network *if* we know that a vertex will be added. $A^v(\cdot)$ may depend on the properties of the whole network in a given time step.

2. $D^v(\cdot)$ describes what kind of vertex is deleted from the network *if* we know that a vertex will be deleted. Again, $D^v(\cdot)$ may depend on the properties of the network in the current time step.

3. $E(\cdot)$ describes which event happens next: edge deletion, edge addition, vertex deletion or vertex addition. This kernel function may depend on the properties of the network in the current time step.

These five kernel functions together serve as the most general kernel-based network evolution framework.

3.2 The frequentist solution

3.2.1 Citation networks

3.2.1.1 The scoring model

The first method we develop for the inverse problem follows a simple scoring model. From the model framework it is clear that vertex types cited many times should have a higher kernel function value compared to rarely cited vertices. (But note that this is only a rule of thumb and not necessarily always literally true.)

The scoring model considers each edge as an independent experiment and for each experiment gives a non-negative score for all vertex types. Vertex types not cited get zero score. Then the estimated kernel function values for each vertex type are obtained as the average of the scores. The assigned score for a given edge should not be always the same constant amount, since we want to assign higher scores for edges which were added when the competition was bigger. Let us give a simple example. In time step two there is a single vertex present in the network, it is thus not a big acknowledgement if the newly added vertex cites this single existing vertex, obviously a smaller score should be given than the one awarded for a citation when there are many vertices present. Also, if a vertex type is very frequent in the network, then citing it should result a smaller score. Surprisingly, applying these two rules linearly in the scoring is enough to obtain the "right" kernel function values. Let's see this formally.

From Equation 3.1 we can easily obtain the kernel function:

$$A(x) = \frac{P[c(e,x)=1]S(t(e))}{N(t(e),x)}. \tag{3.11}$$

An observed edge e either cites an x-vertex or not: if not, then the correct estimation for $P[c(e,x) = 1]$ is zero, thus the approximation, the score for $A(x)$ is also zero. In the other case the approximation is

$$\bar{A}_e(x) = \frac{S(t(e))}{N(t(e), x)}, \tag{3.12}$$

where $\bar{A}_e(x)$ reads as "the approximation of $A(x)$ based on edge e". Taking the average of the scores we get

$$\bar{A}(x) = \frac{1}{|E_x|} \sum_{e \in i(x)} \frac{S(t(e))}{N(t(e), x)}. \tag{3.13}$$

Note that we don't take the average for all e edges as it might happen that in graph $\mathbb{G}(t(e))$ there is no x-vertex and $N(t(e), x)$ is zero. E_x is the subset of the edges for which $\mathbb{G}(t(e))$ contains at least one x-vertex. Obviously, we cannot make experiments for x if it is not represented in the network. $i(x)$ is the set of edges citing x vertices, for other edges the score is zero.

The only problem to solve when applying the frequentist method is that we don't know $S(t)$, as it is a function of $A(\cdot)$, the kernel function which we want to measure. We can, however, apply an iterative approach:

1. We assume that $A_0(x)$ (the 0-th approximation of $A(\cdot)$ is the same for all x, i.e. $A(x) = 1$ and determine $S_0(t)$ accordingly: $S_0(t(e)) = |V(\mathbb{G}(t(e)))|$, the number of vertices in the graph in time step $t(e)$.

2. We use $S_k(t(e))$ to determine $A_{k+1}(x)$ for all x via the frequentist method and calculate $S_{k+1}(t(e))$ from it.

3. We norm the A_{k+1} vector using a suitable vector norm. E.g. we divide by the largest $A(\cdot)$ element. This does not essentially change the kernel function and allows to handle numerical problems.

4. We repeat the previous two steps until $A_k(x)$ and $A_{k+1}(x)$ are closer to each other than a predefined δ threshold.

If this procedure converges, then we obtain a consistent solution for $A(\cdot)$. We will show that the procedure always converges to the unique solution if the network fulfills certain minimal requirements.

3.2.1.2 Convergence and uniqueness

It is not very difficult to prove that the procedure discussed in the previous section converges. For this we write the two substeps of updating $A_{k+1}(x)$

and $S_{k+1}(t)$ into a single equation. We need some more notation. First, let $N^{t(e)} = |V(\mathbb{G}(t(e)))|$ for simplicity. We denote the ratio of x-vertices at time step t by p_x^t, obviously, $\sum_x p_x^t = 1$ for all t. Finally, the set of edges citing x-vertices is denoted by $i(x)$. (More precisely, x-vertices at the time of the citation, as the vertex types may change in time.) Calculating $S_{k+1}(t)$ is done by

$$S_{k+1}(t(e)) = N^{t(e)} \sum_{j=1}^{n} p_j^{t(e)} A_k(j). \tag{3.14}$$

After doing one step of the frequentist estimation we have

$$A_{k+1}(x) = \frac{1}{|E_x|} \sum_{j=1}^{n} \sum_{e \in i(x)} \frac{p_j^{t(e)}}{p_x^{t(e)}} A_k(j). \tag{3.15}$$

The last equation gives a linear transformation of A_k to obtain A_{k+1}. This is obvious if we rewrite it in the form:

$$\begin{array}{rcccccc}
A_{k+1}(1) & = & \frac{1}{|E_1|} \sum_{e \in i(1)} \frac{p_1^{t(e)}}{p_1^{t(e)}} A_k(1) & + & \cdots & + & \frac{1}{|E_1|} \sum_{e \in i(1)} \frac{p_n^{t(e)}}{p_1^{t(e)}} A_k(n) \\
A_{k+1}(2) & = & \frac{1}{|E_2|} \sum_{e \in i(2)} \frac{p_1^{t(e)}}{p_2^{t(e)}} A_k(1) & + & \cdots & + & \frac{1}{|E_2|} \sum_{e \in i(2)} \frac{p_n^{t(e)}}{p_2^{t(e)}} A_k(n) \\
\vdots & & & & & & \\
A_{k+1}(n) & = & \frac{1}{|E_n|} \sum_{e \in i(n)} \frac{p_1^{t(e)}}{p_n^{t(e)}} A_k(1) & + & \cdots & + & \frac{1}{|E_n|} \sum_{e \in i(n)} \frac{p_n^{t(e)}}{p_n^{t(e)}} A_k(n)
\end{array} \tag{3.16}$$

This equation is very similar to the 'power method' which can be used to calculate the leading eigenvector of a matrix in many cases. Let's see briefly how the power method works. If M is a square matrix then a numerical method for calculating its leading eigenvector (i.e. the one corresponding to the largest eigenvalue in absolute value) is the iteration:

$$b_{k+1} = \frac{M b_k}{\|M b_k\|} \tag{3.17}$$

with some b_0 starting vector. It can be proven that the b_k sequence contains a subsequence which converges to the leading eigenvector, provided the following two assumptions are fulfilled:

1. The eigenvalue with the largest absolute value – the dominant eigenvalue – is unique. Note that this also implies that it must be real, as complex eigenvalues appear in conjugate pairs having the same absolute value.

2. The starting vector b_0 has a nonzero component in the direction of the eigenvector associated with the dominant eigenvalue. This assumption can be easily fulfilled if one can choose b_0 freely: $b_0 = (1, 1, \ldots, 1)^{\mathrm{T}}$ is fine.

Our matrix is somewhat special in the respect that all its elements are non-negative. Let us for now assume that all the elements are positive, because this has important consequences for the power method.

If the elements of the matrix are positive then according to the Perron-Frobenius theorem it is true that

1. there is a unique (real) dominant eigenvalue,
2. the eigenvector associated with the dominant eigenvector is all positive.
3. In addition, having a real dominant eigenvalue ensures that the b_k series itself (and not only a subseries) converges to the leading eigenvector.

Note that the convergence of the method does not depend on the vector and matrix norm chosen, although they need to be compatible. This means that the normalization step can be either carried out via dividing $A_{k+1}(\cdot)$ by its maximum element or the sum of all elements. Either way, the same kernel function is obtained, the only difference is that $A(x) = 1$ is set for some x when the first norm is used and $\sum_x A(x) = 1$ is fixed in the second case.

Let's take a look at the all positive matrix assumption. In our case this requirement means that for every pair of vertex types there must be a case when the first vertex type is cited and the second vertex type is present in the network and vice versa. All pairs of vertex types have to be measured against each other. Note that this 'representation restriction' is sufficient for the convergence, but it is an open question whether it is required, too.

With this we have proven that the proposed iteration method always converges to a positive $A(\cdot)$ function provided that the representation restriction is fulfilled. Moreover, we've also proven that the solution of the iteration is unique. To summarize, the following theorem is true:

Theorem 3.1 *Starting from* $[1, 1, \ldots, 1]$, *the frequentist method, as defined by Eq. (3.15) converges to a positive vector if the sufficient representation restriction is fulfilled.*

3.2.2 General networks

It is not difficult to see that the frequentist solution can be applied to non-citation growing networks the same way. Recall that for a non-citation network we have

$$S(t(e)) = \sum_{i,j}^{n} N(t(e),i,j)A(i,j), \tag{3.18}$$

and thus

$$\begin{aligned}\bar{A}(x,y) =& \frac{1}{|E_{xy}|} \sum_{e\in i(x,y)} \frac{S(t(e))}{N(t(e),x,y)} = & (3.19)\\ & \frac{1}{|E_{xy}|} \sum_{i,j} \sum_{e\in i(x,y)} \frac{N(t(e),i,j)}{N(t(e),x,y)} A(i,j), & (3.20)\end{aligned}$$

where E_{xy} is the set of edges e for which there is a possibility to connect an x-vertex to a y-vertex in time step $t(e)$, as usual it is symmetric for undirected networks, and $i(x,y)$ is the set of edges connecting x-vertices to y-vertices, this, too, is symmetric for undirected networks.

This is the same linear transformation as for citation networks, but this time we have n^2 variables instead of n:

$$\begin{aligned}
A_{k+1}(1,1) &= \frac{1}{|E_{11}|}\sum_{e\in i(1,1)} \frac{N_{1,1}^{t(e)}}{N_{1,1}^{t(e)}} A_k(1,1) + \cdots + \frac{1}{|E_{11}|}\sum_{e\in i(1,1)} \frac{N_{n,n}^{t(e)}}{N_{1,1}^{t(e)}} A_k(n,n)\\
A_{k+1}(1,2) &= \frac{1}{|E_{12}|}\sum_{e\in i(1,2)} \frac{N_{1,1}^{t(e)}}{N_{1,2}^{t(e)}} A_k(1,1) + \cdots + \frac{1}{|E_{12}|}\sum_{e\in i(1,2)} \frac{N_{n,n}^{t(e)}}{N_{1,2}^{t(e)}} A_k(n,n)\\
&\vdots\\
A_{k+1}(n,n) &= \frac{1}{|E_{nn}|}\sum_{e\in i(n,n)} \frac{N_{1,1}^{t(e)}}{N_{n,n}^{t(e)}} A_k(1,1) + \cdots + \frac{1}{|E_{nn}|}\sum_{e\in i(n,n)} \frac{N_{n,n}^{t(e)}}{N_{n,n}^{t(e)}} A_k(n,n).
\end{aligned} \tag{3.21}$$

We used the notation $N_{x,y}^{t(e)} = N(t(e),x,y)$ here.

Using the same reasoning as for citation networks, this iteration always converges to a positive $A(\cdot,\cdot)$ vector if the representation restriction is fulfilled.

3.3 The maximum likelihood solution

The frequentist solution discussed in the previous section is intuitive, we estimate a quantity by the average of the measurement experiments performed for it. However, it does not generally guarantees that the obtained kernel function has maximum goodness for the set of fixed vertex properties.

In this section we introduce a maximum likelihood method for extracting the kernel-function from the network evolution data. Maximum likelihood methods try to find the kernel function which is the *most probable* for a given network. In our framework this means that we explicitly search for the kernel function with the highest goodness value.

3.3.1 Citation networks

The probability that a network $\mathbb{G}$ was generated according to the kernel function $A(\cdot)$ is denoted by $P[A(\cdot)|\mathbb{G}]$. According to Bayes' rule this is

$$P[A(\cdot)|\mathbb{G}] = \frac{P[\mathbb{G}|A(\cdot)]P[A(\cdot)]}{P[\mathbb{G}]}, \tag{3.22}$$

where $P[\mathbb{G}|A(\cdot)]$ is the probability that $A(\cdot)$ generates $\mathbb{G}$, $P[A(\cdot)]$ is the probability of the appearance of the kernel function $A(\cdot)$ and $P[\mathbb{G}]$ is the probability of the appearance of the graph(s) $\mathbb{G}$. Sadly, we generally don't know anything about $P[A(\cdot)]$ and assume that every $A(\cdot)$ is equally probable. Here an "Occam's razor" assumption could be used to favor simpler kernel functions, but it is unclear how to do this quantitatively, so we decided to drop it.

$P[\mathbb{G}]$ is only a constant for our concerns, so it turns out that we need to maximize $P[\mathbb{G}|A(\cdot)]$, which is the same as finding the kernel function with maximum goodness. Well, almost. The goodness of a kernel function is not exactly the probability that it generates the network, but the probability that it can guess the outcome of all edge experiments correctly. If we assume that these experiments are independent then this is the same as the probability that the kernel generates the network in exactly the same way as it happened during its evolution.

We need to find

$$\max_{A(\cdot)} \prod_e \frac{A(x_e)}{S(t(e))}. \tag{3.23}$$

If we want to maximize in the space of all possible kernel functions, then the solution is trivial. The property vector includes $t(v)$ and the time step of the citing edge $t(e)$. The kernel function is defined to be one if e cites v

and zero otherwise. This means that e cites vertex v with probability 1 and obviously results the maximum goodness. (We assumed a total ordering on the edges here, an arbitrary compatible total ordering can be defined if the original edge ordering were partial.)

It makes more sense to fix the property vectors and search for the best kernel function based on these fixed vertex types. From now on we focus on the problem of finding the best kernel function values assuming the property vectors are already fixed.

3.3.1.1 The maximization problem

We first define the function to be maximized for citation networks. Recall, that the S normalization factor is

$$S(t(e)) = N^{t(e)} \sum_{i=1}^{n} p_i^{t(e)} A(i). \tag{3.24}$$

Let us denote the number of citations to i-vertices by M_i. Then we need to maximize

$$\prod_e \frac{A(x_e)}{S(t(e))} = \prod_{i=1}^{n} A(i)^{M_i} \prod_e [\sum_{i=1}^{n} p_i^{t(e)} A(i)]^{-1}. \tag{3.25}$$

We omitted the $N^{t(e)}$ factors here, as they don't effect the position of the maximum value.

3.3.1.2 Existence of the solution

Theorem 3.2 *The target function (3.25) has minimum and maximum over the set of non-negative A vectors.*

As the kernel function has no scale, we can restrict ourselves to examine only kernel functions which satisfy $\sum_i A(i) = 1$, where i goes over all vertex types. Then the allowed kernel functions form a compact set as it is closed and bounded. Any continuous function on a compact set has a maximum and a minimum according to the basic theorem of calculus, by Bolzano and Weierstrass. The target function is obviously continuous in all variables (all $A(\cdot)$), thus the existence of the solution follows from the Bolzano-Weierstrass theorem. ∎

3.3.1.3 Uniqueness of the solution

In practice it is better to work with the logarithm of the target function, this is

$$\sum_{i=1}^{n} M_i \log A(i) - \sum_{e} \log[\sum_{i=1}^{n} p_i^{t(e)} A(i)]. \tag{3.26}$$

The partial derivatives according to all $A(k)$ must be zero in order to have a maximum value:

$$\frac{M_k}{A(k)} = \sum_{e} \frac{p_k^{t(e)}}{\sum_{i=1}^{n} p_i^{t(e)} A(i)}, \qquad 1 \le k \le n, \tag{3.27}$$

which can be written as a fixed-point equation form as

$$A(k) = M_k \Big[\sum_{e} \frac{p_k^{t(e)}}{\sum_{i=1}^{n} p_i^{t(e)} A(i)} \Big]^{-1}, \qquad 1 \le k \le n. \tag{3.28}$$

It gives interesting insight if we rewrite these equations in the form

$$\sum_{e} \frac{p_k^{t(e)} A(k)}{\sum_{i=1}^{n} p_i^{t(e)} A(i)} = M_k, \qquad 1 \le k \le n. \tag{3.29}$$

The left hand side of (3.29) is the expected number of citations to k-vertices based on a given $A(\cdot)$ kernel function, the right hand side is the observed number of citations. The optimal kernel function requires that these two are exactly the same.

It is true that the target function is minimal if $A(k) = 0$ for some k and the minimum value is zero. This means that if $A(k) > 0$ then all singular points (the points in which all partial derivatives are zero) are either maximum points or saddle points. We will prove that for any two $A^1(\cdot)$ and $A^2(\cdot)$ singular points satisfying (3.29), it is true that $A^1(\cdot) = cA^2(\cdot)$ with some $c > 0$ real number. From this it follows that every such point is a maximum point, otherwise no maximum point would exist.

Let us assume that the A^1 and A^2 kernel functions satisfy (3.29), from this we have

$$\sum_{e} \frac{p_k^{t(e)} A^1(k)}{\sum_{i=1}^{n} p_i^{t(e)} A^1(i)} = \sum_{e} \frac{p_k^{t(e)} A^2(k)}{\sum_{i=1}^{n} p_i^{t(e)} A^2(i)}, \qquad 1 \le k \le n, \tag{3.30}$$

and by denoting the two normalization factors by $S^1(e)$ and $S^2(e)$ we get

$$\sum_{e} \frac{p_k^{t(e)} A^1(k) S^2(e) - p_k^{t(e)} A^2(k) S^1(e)}{S^1(e) S^2(e)} = 0, \qquad 1 \le k \le n. \tag{3.31}$$

Some simple algebra leads to

$$\sum_e \frac{p_k^{t(e)} A^1(k) \sum_{i=1}^n p_i^{t(e)} A^2(i) - p_k^{t(e)} A^2(k) \sum_{i=1}^n p_i^{t(e)} A^1(i)}{S^1(e)S^2(e)} =$$

$$= \sum_e \frac{p_1^{t(e)} p_k^{t(e)} A^1(k)A^2(1) + \cdots + p_n^{t(e)} p_k^{t(e)} A^1(k)A^2(n)}{S^1(e)S^2(e)} -$$

$$- \sum_e \frac{p_1^{t(e)} p_k^{t(e)} A^1(1)A^2(k) + \cdots + p_n^{t(e)} p_k^{t(e)} A^1(n)A^2(k)}{S^1(e)S^2(e)} =$$

$$= (A^1(k)A^2(1) - A^2(k)A^1(1)) \sum_e \frac{p_1^{t(e)} p_k^{t(e)}}{S^1(e)S^2(e)} + \cdots +$$

$$+ (A^1(k)A^2(n) - A^2(k)A^1(n)) \sum_e \frac{p_n^{t(e)} p_k^{t(e)}}{S^1(e)S^2(e)} = 0 \quad (3.32)$$

We assume that

$$\sum_e \frac{p_i^{t(e)} p_j^{t(e)}}{S^1(e)S^2(e)} > 0, \qquad 1 \le i, j \le n, \quad (3.33)$$

then the sum in (3.32) can be zero in two different ways.

1. All terms are zero, i.e.

   $$A^1(k)A^2(i) = A^2(k)A^1(i), \qquad 1 \le i \le n, \quad (3.34)$$

   and from here

   $$A^1(i) = cA^2(i), \qquad 1 \le i \le n. \quad (3.35)$$

2. There is at least one negative term. There exists $1 \le i \le n$, $i \ne k$, such that $A^1(k)A^2(i) - A^2(k)A^1(i) < 0$,

   $$\frac{A^2(i)}{A^1(i)} < \frac{A^2(k)}{A^1(k)} \quad (3.36)$$

   However, the procedure can be done for all $1 \le k \le n$, so for every $1 \le k \le n$ there must be an index $1 \le i \le n$, $i \ne k$ such that equation (3.36) holds. This is clearly contradiction.

The uniqueness of the solution is thus proven. It was also shown, that any singular point $A(i) > 0$, $1 \le i \le n$ implies that $A(\cdot)$ is a kernel function with maximum goodness. In other words, the target function has a single global (and local) maximum if the allowed solutions are restricted either by fixing $A(x) = 1$ for some x or by fixing $\sum_i A(i) = 1$. This implies that any optimization algorithm, which is able to find a *local* maximum of a continuous non-linear function in finite time, is appropriate for solving the maximum likelihood problem.

The assumption given in equation (3.33) means that for every pair of vertex types, there must be a time step when they are both present in the network. This is the sufficient 'representation restriction' for the maximum likelihood method. The following theorem was thus proved:

Theorem 3.3 *The target function (3.25) has a unique maximum over the set of positive A vectors (unique, apart from normalization), if the representation restriction of the maximum likelihood method is true.*

3.3.1.4 Stationary vertex type distribution

In this section we deal with a simple subclass of networks for which the frequentist method and the maximum likelihood method give the same result. In these networks the distribution of vertex types is stationary in time. We use the following notation: p_i is the ratio of i-vertices in the network ($1 \leq i \leq n$), there are n different vertex types. Obviously it is true that

$$\sum_{i=1}^{n} p_i = 1. \tag{3.37}$$

The number of citations to an i-vertex is denoted by M_i, the total number of edges is M. $N^{t(e)}$ is the number of vertices in the network in time step $t(e)$, the total number of vertices is N. In stationary networks it is true that

$$S(t(e)) = N^{t(e)}[p_1 A(1) + p_2 A(2) + \cdots + p_n A(n)] = N^{t(e)} \sum_{i=1}^{n} p_i A(i). \tag{3.38}$$

For simplicity we will sometimes use the notation

$$p^* = \sum_{i=1}^{n} p_i A(i). \tag{3.39}$$

Thus we need to maximize

$$\prod_e \frac{A(x_e)}{S(t(e))} = \frac{A(x_1)}{S(t(1))}\frac{A(x_2)}{S(t(2))}\cdots\frac{A(x_M)}{S(t(M))} = \tag{3.40}$$

$$= \prod_i^n A(i)^{M_i} \frac{1}{N^{t(1)}p^*}\cdots\frac{1}{N^{t(M)}p^*} = \tag{3.41}$$

$$= [\prod_i^n A(i)^{M_i}][\prod_e \frac{1}{N^{t(e)}}][\frac{1}{p^*}]^M. \tag{3.42}$$

As $N^{t(e)}$ are fixed parameters, we can omit them, and the quantity to optimize is

$$[\prod_i^n A(i)^{M_i}][\sum_{i=1}^n p_i A(i)]^{-M}. \tag{3.43}$$

Let us take the logarithm first:

$$\sum_{i=1}^n M_i \log A(i) - M \log \sum_{i=1}^n p_i A(i). \tag{3.44}$$

For the maximum it is required that partial derivatives according to all $A(k)$ $(1 \leq k \leq n)$ are zero, this gives us n equations:

$$\frac{M_k}{A(k)} = \frac{M p_k}{\sum_{i=1}^n p_i A(i)}, \qquad 1 \leq k \leq n, \tag{3.45}$$

which can be written as

$$p_1 A(1) + \cdots + p_k(1 - \frac{M}{M_k})A(k) + \cdots + p_n A(n) = 0, \qquad 1 \leq k \leq n. \tag{3.46}$$

We have a system of n linear equations:

$$\begin{array}{ccccccccc} p_1(1-\frac{M}{M_1})A(1) & + & p_2A(2) & + & \cdots & + & p_nA(n) & = 0 \\ p_1A(1) & + & p_2(1-\frac{M}{M_2})A(2) & + & \cdots & + & p_nA(n) & = 0 \\ \vdots & & & & & & & \\ p_1A(1) & + & p_2A(2) & + & \cdots & + & p_n(1-\frac{M}{M_n})A(n) & = 0. \end{array} \tag{3.47}$$

If we subtract all other lines from the first line we get

$$\begin{aligned} A(2) &= \frac{p_1}{p_2}\frac{M_2}{M_1}A(1) \\ &\vdots \\ A(n) &= \frac{p_1}{p_k}\frac{M_k}{M_1}A(1), \end{aligned} \tag{3.48}$$

which means that after fixing $A(1) = 1$ (or some other arbitrary positive value), we can calculate all values of the kernel function. The maximum likelihood problem can be easily solved for stationary networks.

It can be shown that if the vertex type distribution is stationary then the frequentist method is equivalent to the maximum likelihood method.

Equation (3.16) gives the iteration procedure performed for the frequentist method in the general case. If we now assume that the distribution of the vertex types is stationary, then it reduces to

$$\begin{aligned}
A_{k+1}(1) &= \frac{M_1}{M}\frac{p_1}{p_1}A_k(1) + \frac{M_1}{M}\frac{p_2}{p_1}A_k(2) + \cdots + \frac{M_1}{M}\frac{p_n}{p_1}A_k(n)\\
A_{k+1}(2) &= \frac{M_2}{M}\frac{p_1}{p_2}A_k(1) + \frac{M_2}{M}\frac{p_2}{p_2}A_k(2) + \cdots + \frac{M_2}{M}\frac{p_n}{p_2}A_k(n)\\
&\vdots\\
A_{k+1}(n) &= \frac{M_n}{M}\frac{p_1}{p_n}A_k(1) + \frac{M_n}{M}\frac{p_2}{p_n}A_k(2) + \cdots + \frac{M_n}{M}\frac{p_n}{p_n}A_k(n).
\end{aligned} \tag{3.49}$$

As discussed in Section 3.2.1.2., if the representation restriction for the frequentist method is true, then this iteration converges to the leading eigenvector of its matrix and it is true that

$$\begin{aligned}
A(1) &= \frac{M_1}{M}\frac{p_1}{p_1}A(1) + \frac{M_1}{M}\frac{p_2}{p_1}A(2) + \cdots + \frac{M_1}{M}\frac{p_n}{p_1}A(n)\\
A(2) &= \frac{M_2}{M}\frac{p_1}{p_2}A(1) + \frac{M_2}{M}\frac{p_2}{p_2}A(2) + \cdots + \frac{M_2}{M}\frac{p_n}{p_2}A(n)\\
&\vdots\\
A(n) &= \frac{M_n}{M}\frac{p_1}{p_n}A(1) + \frac{M_n}{M}\frac{p_2}{p_n}A(2) + \cdots + \frac{M_n}{M}\frac{p_n}{p_n}A(n).
\end{aligned} \tag{3.50}$$

If we divide line i by M_i, multiply by Mp_i and subtract the right side we get a very similar equation to (3.47), and from this the same (3.48) solution follows, the same way, by subtracting all other lines from the first one.

3.3.1.5 Two vertex types

For two vertex types the general non-stationary maximum-likelihood problem can be solved by using a fixed point equation. Let us recall that the solution of the problem is the kernel function satisfying the fixed-point equation (3.28). For two vertex types, if we fix $A(1) = 1$, then the fixed-point equation reads as

$$A(2) = M_2\Big[\sum_e \frac{p_2^{t(e)}}{p_1^{t(e)} + p_2^{t(e)}A(2)}\Big]^{-1}. \tag{3.51}$$

From now on we assume that $A(1) \geq A(2)$. If this turns out to be not true, then we simply switch the 1-vertices and the 2-vertices. It is actually quite simple to determine whether $A(1) \geq A(2)$ or the opposite is true:

Lemma 3.1 *In a network with two vertex types, it is true that*

$$A(1) \geq A(2) \text{ if and only if } \sum_e p_1^{t(e)} \leq M_1 \quad (\text{and } \sum_e p_2^{t(e)} \geq M_2). \tag{3.52}$$

1. $A(1) \geq A(2)$; since $A(1) = 1$ is fixed this means $A(2) \leq 1$. Since $p_1^{t(e)} + p_2^{t(e)} = 1$ holds for all e, $p_1^{t(e)} + p_2^{t(e)} A(2) \leq 1$ also holds for all e. Thus in the fixed point it is required that

$$M_1 = \sum_e \frac{p_1^{t(e)}}{p_1^{t(e)} + p_2^{t(e)} A(2)} \geq \sum_e p_1^{t(e)}. \tag{3.53}$$

2. $\sum_e p_1^{t(e)} \leq M_1$. In the fixed point it is required that

$$\sum_e p_1^{t(e)} \leq M_1 = \sum_e \frac{p_1^{t(e)}}{p_1^{t(e)} + p_2^{t(e)} A(2)}. \tag{3.54}$$

   Since $p_1^{t(e)} + p_2^{t(e)} = 1$ for all e, it is either true that $p_1^{t(e)} + p_2^{t(e)} A(2) > 1$ for all e, or $p_1^{t(e)} + p_2^{t(e)} A(2) \leq 1$ holds for all e. Clearly, the latter is required to satisfy (3.54), and this implies $A(1) = 1 \geq A(2)$.

$\blacksquare$

Let us examine the

$$f(x) := M_2 \Big[\sum_e \frac{p_2^{t(e)}}{p_1^{t(e)} + p_2^{t(e)} x} \Big]^{-1}. \tag{3.55}$$

function.

Theorem 3.4 *f possesses the following properties:*

1. *For the unique x^* fixed point of f it is true that $x^* \leq 1$.*

2. *f is strictly monotone increasing.*

3. *f is concave.*

4. *If $x < x^*$ then $f(x) < x^*$ where $x^* = f(x^*)$ is the unique fixed point of f.*

5. *If $x > x^*$ then $f(x) > x^*$.*

6. *If $x < x^*$ then $x^* - f(x) < x^* - x$.*

7. *If $x > x^*$ then $f(x) - x^* > x - x^*$.*

8. *$|f(x) - x^*| < |x - x^*|$ for all $x \in [0, 1]$.*

9. *The $x_{n+1} := f(x_n)$ fixed point iteration converges to the unique fixed point from every $x_0 \in [0, 1]$ starting point.*

1. This is simply the consequence of the assumption that $A(1) \geq A(2)$.

2. It is easy to see that it's derivative is strictly positive:

$$f'(x) = M_2 \frac{\sum_e \left[\frac{p_2^{t(e)}}{p_1^{t(e)}+p_2^{t(e)}x}\right]^2}{\left[\sum_e \frac{p_2^{t(e)}}{p_1^{t(e)}+p_2^{t(e)}x}\right]^2} > 0, \tag{3.56}$$

   as the two sums are both always positive and M_2 is positive too. (We don't deal with the uninteresting $M_2 = 0$ case.)

3. The second derivative of f is strictly negative, so the function is strictly concave:

$$\begin{aligned} f''(x) = M_2 & \frac{-2\left[\sum_e \left(\frac{p_2^{t(e)}}{p_1^{t(e)}+p_2^{t(e)}x}\right)^3\right]\left[\sum_e \frac{p_2^{t(e)}}{p_1^{t(e)}+p_2^{t(e)}x}\right]^2}{\left[\sum_e \frac{p_2^{t(e)}}{p_1^{t(e)}+p_2^{t(e)}x}\right]^4} + \\ & + \frac{-2\left[\sum_e \left(\frac{p_2^{t(e)}}{p_1^{t(e)}+p_2^{t(e)}x}\right)^3\right]^2\left[\sum_e \frac{p_2^{t(e)}}{p_1^{t(e)}+p_2^{t(e)}x}\right]}{\left[\sum_e \frac{p_2^{t(e)}}{p_1^{t(e)}+p_2^{t(e)}x}\right]^4}, \end{aligned} \tag{3.57}$$

   which is

$$f''(x) = -2M_2 \frac{\sum_e \left(\frac{p_2^{t(e)}}{p_1^{t(e)}+p_2^{t(e)}x}\right)^3}{\left[\sum_e \frac{p_2^{t(e)}}{p_1^{t(e)}+p_2^{t(e)}x}\right]^2} - 2M_2 \frac{\left[\sum_e \left(\frac{p_2^{t(e)}}{p_1^{t(e)}+p_2^{t(e)}x}\right)^2\right]^2}{\left[\sum_e \frac{p_2^{t(e)}}{p_1^{t(e)}+p_2^{t(e)}x}\right]^3} < 0. \tag{3.58}$$

The inequality holds, except in the uninteresting case when $p_2^{t(e)} = 0$ for all e.

4. Since f is strictly monotone increasing, $x < x^*$ implies $f(x) < f(x^*) = x^*$.

5. Since f is strictly monotone increasing, $x > x^*$ implies $f(x) > f(x^*) = x^*$.

6. $x < x^*$ and $f(0) > 0$ imply $x < f(x)$ and this in turn implies $x^* - f(x) < x^* - x$.

7. $x > x^*$, $f(x^*) = x^*$ and $f(0) > 0$ implies $x > f(x)$ and this in turn implies $f(x) - x^* < x - x^*$.

8. The previous four properties simply ensure $|f(x) - x^*| < |x - x^*|$ for all $x \in [0, 1]$.

9. Since the fixed point is unique, the previous property ensures the convergence of the fixed point operation from all starting point. ∎

3.3.1.6 Convergence of the fixed-point equation

Previously we've shown that the general maximum likelihood problem can be solved using any optimization method capable of finding a local maximum of a non-linear continuous function. We've also shown that for two vertex types the simple fixed point iteration always converges to the solution. It is a natural question whether we can solve the general problem with the same simple fixed point iteration, as defined in Eq. (3.28).

Although we tried several different approaches, we couldn't prove that the simple fixed point equation is convergent in the general case. Based on numeric simulations we present the following conjectures as open problems:

Conjecture 3.1 *The (3.28) fixed-point equation is convergent from all starting vectors, if after each iteration we normalize $A(\cdot)$ according to $\|A(\cdot)\| = 1$.*

Conjecture 3.2 *The (3.28) fixed-point equation is convergent from all starting vectors. (Even without the normalization in the previous conjecture.)*

Before the third conjecture we need to define quasi-contractive maps.

Definition 3.1 *Let $T : M \to M$ a mapping of a metric space (M, d) into itself. A mapping T is called a quasi-contraction if and only if $d(Tx, Ty) \leq q \cdot \max\{d(x, y), d(x, Tx), d(y, Ty), d(x, Ty), d(y, Tx)\}$ for some $q < 1$ and all $x, y \in M$.*

Conjecture 3.3 *The function defined in equation (3.28) is quasi-contraction. (With out without the normalization step.)*

Conjectures 3.1 and 3.2 follow from Conjecture 3.3 according to the work of Ciric [1974].

3.3.1.7 The required representation restriction

There is a natural restriction on the networks for which the maximum likelihood method can be used: for every vertex type there must be an edge which does not cite it at a time step when it is present in the network. In other words, if a vertex type is cited by k edges then it must be present at least in $k + 1$ time steps. This is natural, because if a vertex type is cited *every time* it is present in the network that would imply that it is infinitely more attractive than all other vertex types.

The representation restriction can be seen easily from the usual maximization derivation. In the general non-stationary case we want to maximize

$$\Big[\prod_{i=1}^{n} A(i)^{M_i}\Big] \sum_{e} \Big[\sum_{j=1}^{n} p_j^{t(e)} A(j)\Big]^{-1}. \tag{3.59}$$

This leads to the usual equations

$$M_k = \sum_{e} \frac{p_k^{t(e)} A(k)}{\sum_{j=1}^{n} p_j^{t(e)} A(j)}, \qquad 1 \leq k \leq n. \tag{3.60}$$

If 1-vertices are always cited whenever they are present in the system and we set $A(1) = 1$ then we can write the equation for $A(1)$ as

$$M_1 = \sum_{e \in E_1} \frac{p_1^{t(e)}}{p_1^{t(e)} + \sum_{j=2}^{n} p_j^{t(e)} A(j)}, \tag{3.61}$$

where E_1 is the set of edges citing 1-vertices and other edges can be eliminated from the sum because if an edge is not citing a 1-vertex then there is no 1-vertex in the network and $p_1^{t(e)} = 0$ if $e \notin E_1$.

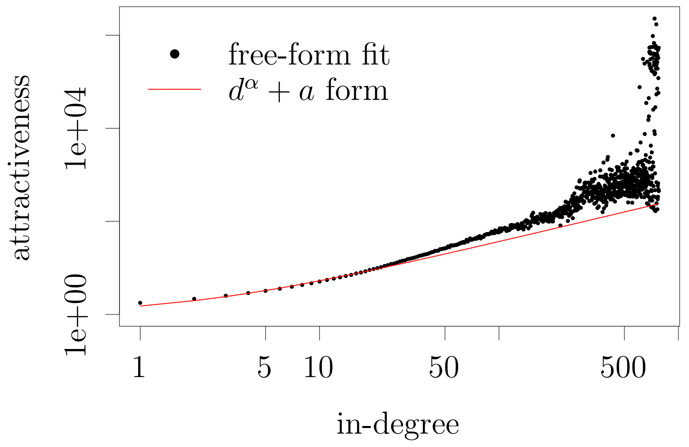


Figure 3.4: The measured kernel function of the US patent citation network, based on in-degree. The red line is the maximum likelihood fitted $A(d) = d^\alpha + a$ form. The axes are logarithmic, so $d = 0$ is not included in the plot.

This sum has M_1 terms and every term is less than 1, thus the equation has no solution. It is possible, however, to give an arbitrarily precise approximation by setting all $A(i)$, except for $A(1)$, small enough. In other words, all $A(i)$ must be infinitesimally small compared to $A(1)$, or $A(1)$ must be infinite compared to all other $A(i)$. (The latter can be directly derived as well by setting $A(i) = 1$ for some $i \neq 1$.)

The representation restriction is required in order to have a finite solution for the maximum likelihood problem but it is an open question whether it is sufficient.

3.3.2 Kernel functions with predefined shape

While the maximum likelihood method provides an elegant and practical solution to extract the kernel function from the evolution of a network, it is often desired to give the kernel function in a simple functional form. See Fig. 3.4 as an example. Here the kernel function depends on the in-degree of the vertices and we measured it on the US patent citation network (see Section 4.2.2).

While it is almost apparent, that the kernel function can be well fitted with the $A(d) = d^\alpha + a$ form, it is not quite straightforward how the actual fit should be produced. One issue is that the kernel function values for the different vertex types are based on different number of experiments: for low degrees there are very many, for high degrees very few. In the extreme case, we might have a single experiment for a kernel function value. Clearly, weighted fitting should be used. It is also not obvious that some fitting method, e.g. least squares, provides the best fit with respect to the goodness of the fitted function.

This, again, leads us to maximum likelihood fitting, but this time with respect to the parameters of the fitted form. In our example we search for the parameter values α and a which give the highest goodness for $A(d) = d^\alpha + a$. This maximum likelihood problem, however, behaves not as nicely as the free-form kernel function fitting discussed so far. In general, nothing ensures the uniqueness or even the existence of the solution. It might be also difficult to find an optimization method which converges for the particular form and the particular network. It is still worth mentioning this method, however, because—if it works—it gives a straightforward, parameter-free method for fitting a functional form.

By comparing the goodness scores of the free-form function and the fitted form, it is possible to quantitatively give the "error" introduced by the functional form. Again, see Fig. 3.4 for an example.

Having a kernel-function shape with m parameters $q_1, \dots, q_m$ we need to maximize

$$\sum_{i=1}^{n} M_i \log A(q_1, \dots, q_m, x_i) - \sum_e \log[\sum_{i=1}^{n} p_i^{t(e)} A(q_1, \dots, q_m, x_i)]. \tag{3.62}$$

As some optimization methods, like the often used BFGS algorithm [Nocedal and Wright, 1999], can make good use of the partial derivatives of the objective function, it is worth calculating them. For a q_j parameter:

$$\sum_{i=1}^{n} \frac{M_i \partial_{q_j} A(q_1, \dots, q_m, x_i)}{A(q_1, \dots, q_m, x_i)} - \sum_e \frac{\sum_{i=1}^{n} p_i^{t(e)} \partial_{q_j} A(q_1, \dots, q_m, x_i)}{\sum_{i=1}^{n} p_i^{t(e)} A(q_1, \dots, q_m, x_i)}. \tag{3.63}$$

3.3.3 The maximum likelihood method as a scoring model

In Section 3.2 we developed the frequentist method as a simple scoring model: each edge served as an experiment and every vertex type got a score in every experiment. In that method the score was zero if the vertex type was not cited and $S(t(e))/N(t(e), x)$ otherwise. Then the estimated kernel function value for a vertex type was simply the average of its scores.

Now, we give a similar formulation of the maximum likelihood method. We now give a score of

$$\frac{M_k}{M} \frac{S(t(e))}{p_k^{t(e)}} \tag{3.64}$$

to every vertex type present in the network in time step $t(e)$. After "performing" all experiments, we take the *harmonic* mean on the scores for each vertex type. This way the estimated kernel function value for k-vertices is:

$$\bar{A}(k) = \frac{M}{\frac{Mp_k^{t(1)}}{M_k S(t(1))} + \cdots + \frac{Mp_k^{t(M)}}{M_k S(t(M))}} = M_k [\sum_e \frac{p_k^{t(e)}}{S(t(e))}]^{-1}. \tag{3.65}$$

which is exactly the fixed point equation for the solution of the maximum likelihood method.

3.3.4 General networks

It is easy to generalize the maximum likelihood solution to non-citation networks. For these the goodness of the kernel function is defined as

$$\prod_e \frac{A(x_e^*, x_e^{**})}{S(t(e))} = \prod_{i=1}^n \prod_{j=1}^n A(i,j)^{M_{ij}} \prod_e [\sum_{i,j=1}^n N(t(e),i,j) A(i,j)]^{-1}, \tag{3.66}$$

where edge e connected an x_e^*-vertex to an x_e^{**} vertex in the analyzed network and if the network is undirected then the sum and the product also have the condition $i \le j$.

The maximization of (3.66) is equivalent to the citation network case if we create new variables from the (i,j) pairs. Denoting (i,j) by k we have

$$\prod_e \frac{A^*(x_e)}{S(t(e))} = \prod_{k=1}^{n^*} A(k)^{M_k} \prod_e [\sum_{k=1}^{n^*} N(t(e),k) A(k)]^{-1}, \tag{3.67}$$

where $A^*(\cdot)$ is the new kernel function and n^* is the number of new variables. It is $n^* = n \cdot (n-1)$ for directed networks without loop edges and $n^* = (n \cdot (n-1))/2$ for undirected networks.

The existence, uniqueness and all the other properties proven for the citation network case are also valid here.

3.4 Generalizing goodness: conditioning on other network properties

The goodness of a kernel function, as defined in Eq. (3.4), measures the probability that the kernel is able to guess all edge experiments. While this

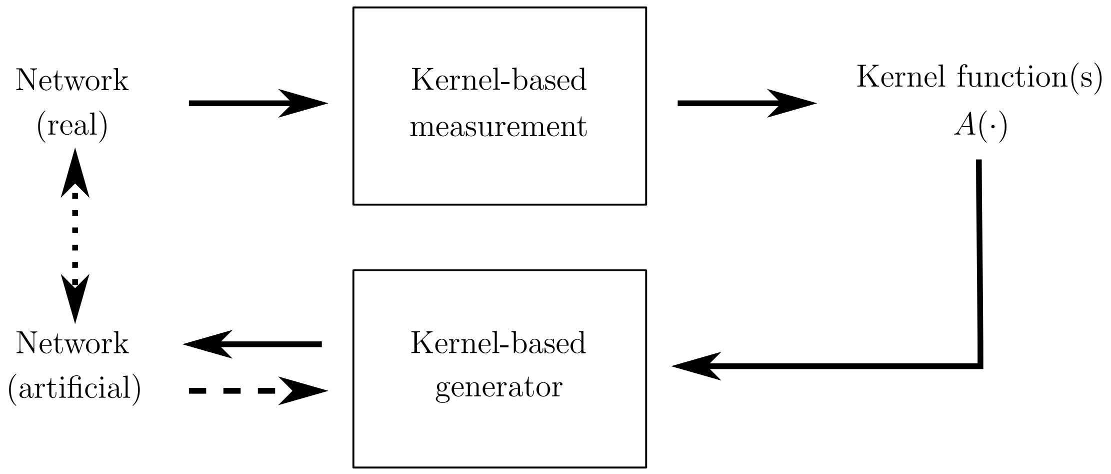


Figure 3.5: Generalizing goodness, sketch of the methodology. Based on the measured kernel function, we generate artificial kernel-based networks, calculate the error by comparing the real and artificial networks (dotted arrow) and generate new artificial networks if necessary (ragged arrow).

is an appropriate definition in many senses, both theoretically and practically, it is not always the best for all purposes.

Let us assume for example that we are interested in the in-degree distribution of our network(s), and want to find the kernel function based on some property vectors which reproduces the in-degree distribution best. Using the vertex in-degree as a property seems to be reasonable, but it is not quite straightforward what other properties we might want to include and even in the case of having only the in-degree, it is not clear that the kernel with the highest goodness is best for the in-degree distribution as well. (We call this goodness, defined as the error in the in-degree distribution or some other network property, "generalized goodness".

We suggest the following heuristic approach. We fit (e.g. with the maximum likelihood method) the network to find the maximum (traditional) goodness kernel. Then, maybe, after also fitting a predefined kernel shape, try to "perturb" the fitted parameters and generate synthetic kernel based networks using the perturbed parameters to see whether they result higher generalized goodness. E.g. it is possible to use some nonlinear optimization method [Nocedal and Wright, 1999] to find the best kernel in terms of generalized goodness. Fig. 3.5 gives a summary of this methodology. We discuss the generation of kernel based networks in Appendix A.

3.5 Time complexity of the algorithms

The frequentist method involves the calculation of the leading eigenvector of a matrix of size n, the number of vertex types appearing during network evolution. There are a number of way to implement this, most of them requires matrix-vector multiplications, with a dense matrix, and this can be done in $O(n^2)$ time.

The matrix itself can be usually created in $O(N + M)$ time, N is the number of vertices in the network, M is the number of edges. The critical part is to keep track of the $p_i^{t(e)}$ probabilities efficiently. In fact, $N_i^{t(e)}$, the number of vertices for all types can be just as well used for the calculations, instead of $p_i^{t(e)}$, no need for norming the probabilities after every change. For some models it is easy to follow $N_i^{t(e)}$. E.g. in the in-degree based model every citing edge changes just one $N_k^{t(e})$, the one belonging to the cited vertex type, plus the addition of a new vertex changes $N_0^{t(e)}$. In other models, like the ones including the age of the vertices it is a bit more difficult as the age of all vertices needs to updated in every time step. By binning the age into larger units we can obtain better running time.

For non-citation networks, we need to keep track of all the possible connections between vertex types. This can be done efficiently by a vector (R) and a matrix (B). The vector simply stores the current number of vertices of a given kind and the matrix gives the already realized connections between all pairs of vertex types. Then $N_{i,j}^{t(e)}$ can be calculated as $R_i R_j - B_{i,j}$ if $i \neq j$ and as $R_i(R_i - 1)/2 - B_{i,i}$ if $i = j$.

For the frequentist method, all measurement algorithms we implemented have linear or closely linear running time if $O(1)$ iterations are performed.

Implementing the maximum likelihood method is quite simple, we can use any optimization method, or the fixed-point iteration. Both require keeping track of the $N_i^{t(e)}$ counts, just like for the other method. The maximum likelihood method has usually linear or closely linear running time, too.

4

Applications

THIS CHAPTER CONTAINS some applications of the methodology discussed so far. Note that at the time performing the applications in Sections 4.1 and 4.2 we did not invent the notion of goodness, Eq. (3.4) and the maximum likelihood method yet (Sec. 3.3), these were performed using the frequentist solution, as discussed in Section 3.2.

4.1 Citation Prediction

The ACM Special Interest Group on Knowledge Discovery and Data Mining organizes a conference each year and together with the conference they also host a data mining competition called KDD Cup. In 2003 the first task of the KDD Cup was to predict the citations to the papers in the high energy physics database. This database contains high energy papers submitted to the arXiv e-print archive between 1992 and July 31, 2003. The deadline for the KDD Cup submission was before April 30, 2003 and the citations made by papers in the next three months had to be predicted. See `http://www.cs.cornell.edu/projects/kddcup/` for more information on the 2003 KDD Cup.

The evaluation of the prediction algorithms was done by considering only papers receiving at least six citations during the period February 1, 2003 – April 30, 2003. For these papers first the target vector, the difference between the citations received between May and August, and between February and May were calculated. The specific task was the prediction of this vector. The error of the prediction was simply defined by the absolute value of the difference of the prediction and the target vector.

While the kernel-based method was not primarily developed for citation prediction, is can be used for that in the following way. We can measure the dynamics (i.e. the $A(\cdot)$ function) of the network up to *now* and assuming that this function will be the same in the future we can simulate the growth of the network according to the measured dynamics and see a possible realization of the network (say) three months later. By generating many realizations and taking the average number of citations a node received in these realizations we can predict the "average" expected evolution of the network.

Note that although we don't use the usual kernel function goodness here (it was not yet defined when this work was done), but the the citation prediction error we use is perfectly equivalent to the goodness.

At the 2003 KDD Cup, the error of the winner algorithm was 1329. The totally random network evolution, where each new node connects to a number of randomly selected nodes yields, on the average, an error of 3463. This value was obtained by averaging hundred totally random realizations. These error values can be used as baselines to place the error of the predictions of our method.

First we measured the $A(\cdot)$ function based on the in-degree of the nodes solely and found that the

$$A(k) = k^{\alpha} + 1 \tag{4.1}$$

form gives a reasonable good fit with the measured data. We fitted this form by a simple weighted least square method and got $\alpha \approx 0.85$. The prediction with this $A(k)$ function yielded an error about $2473.51(\pm 4.39)$. These values were obtained by generating 100 realizations five times, the error is simply the standard deviation of the five predictions.

To evaluate our dynamics measurement method we've calculated predictions with other α exponents as well, and found that the $\alpha = 0.85$ value is very close to the "optimal" exponent, optimal in terms of the error of this prediction, see Fig. 4.1.

Instead of using solely the in-degree as the predictor, now we will also add the age of the nodes, we measure the $A(k, l)$ function, as before k being the in-degree and l being the age of a node. The measured $A(k, l)$ function can be reasonably well fitted by the following form:

$$A(k, l) = (k^{\alpha} + 1)\, l^{-\beta} \ . \tag{4.2}$$

This form assumes that the effect of in-degree and age can be separated, our data supports this assumption. By fitting this form using weighted least square fits we arrive to the exponents: $\alpha \approx 1.14$ and $\beta \approx 1.14$. By using these values in generating possible realizations of the HEP network for the

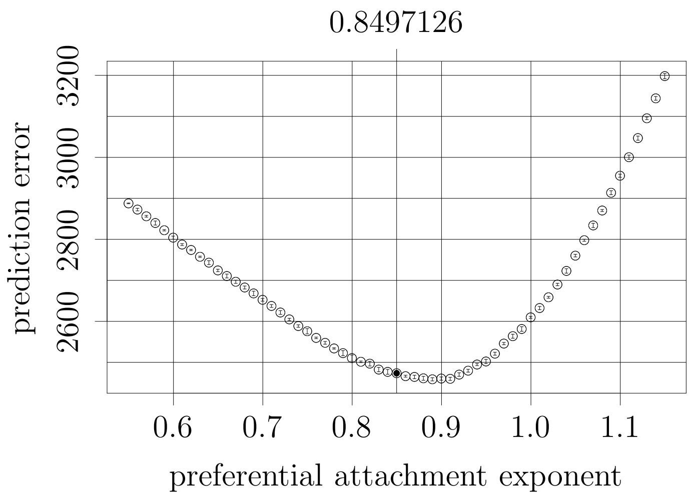
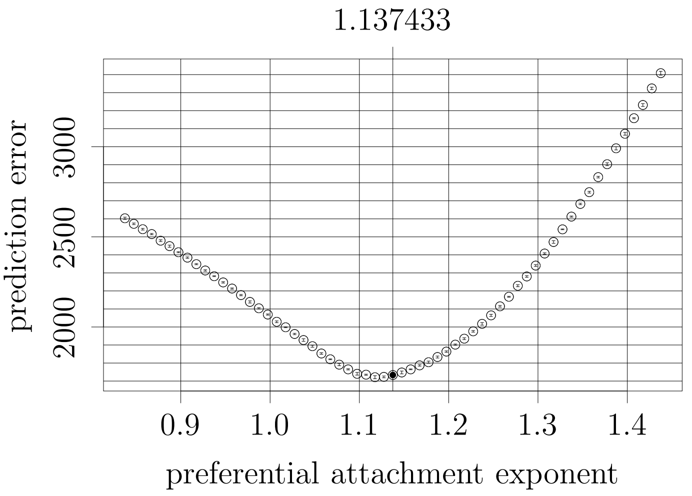


Figure 4.1: Prediction error for different α values in (4.1). The plot was obtained by running five times 100 realizations for each α value, the error bars show the standard deviation of the five predictions. The measured 0.85 exponent is close to the optimal 0.89 value.

prediction we get a prediction error 1732.76 ± 6.19. The fact that this prediction is much better than the "in-degree-only" one, indicates that the age of the nodes makes an important contribution to the dynamics of the evolving network. See Fig. 4.2 for the prediction error of the in-degree and age-based model.

Note that the exponent of the preferential attachment is lower if we don't use the age of the papers as a property, $\alpha_k \approx 0.85$ versus $\alpha_{k,l} \approx 1.14$. This is clearly because in the former the effect of the aging is "built in" into the preferential exponent and since aging works *against* preferential attachment it makes the exponent smaller. Some works suggest that the preferential attachment mechanism can be present even in a network not showing the scale-free degree distribution because there is another, opposite effect working in the system, such as limits for the number of edges a node can acquire or because the nodes lose their "attractiveness" by getting older, i.e. aging, see Amaral et al. [2000]. To our knowledge our work is the first one giving experimental evidence for this assumption.



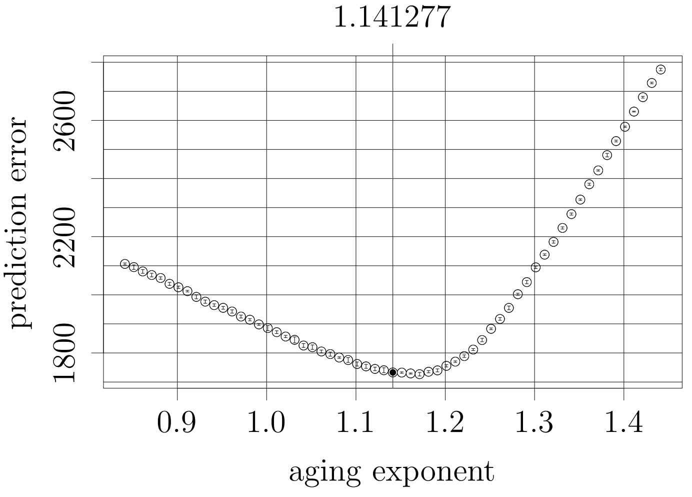


Figure 4.2: Prediction error for different preferential attachment exponents (α, upper plot) and aging exponents (β, lower plot). For both exponents the dynamics measurement method gives solutions close to the optimal ones.

4.2 Identifying changes in the dynamics

In this section we apply the introduced techniques to the United States patent citation network and draw conclusions about its dynamics. We examine whether the trace of the early nineties' legal changes can be found in the dynamics by kernel-based modeling. All measurements discussed in this section were performed using the frequentist method.

4.2.1 Introduction

Innovation plays a key role in economic development and the patent system is intended (and required by the United States Constitution) to promote innovation. The patent system promotes innovation by giving inventors the power to exclude others from using their inventions during the patent term. The power to exclude is a double-edged sword, however, because it benefits the original inventor, but imposes costs on later innovators seeking to build on past inventions. Thus, the proper design of the patent system is an important matter – and a matter of considerable current debate. See, e.g., Jaffe and Lerner [2004], Federal Trade Commission [2003], Merrill et al. [2004]. Advances in computer technology and the availability of large patent databases have recently made it possible to study aspects of the patent system quantitatively. Because patents and the citations between them can be conceptualized as a growing network techniques from statistical physics that have been used in the study of complex networks can be usefully applied to the patent citation network Albert and Barabási [2002], Newman [2003a], Clarkson [2003], Breschi and Lissoni [2004].

The section is organized as follows: In Section 4.2.2 we provide background on the United States patent system and describe the citation data that is used in this study. In Section 4.2.3 we briefly review the kernel-based methodology and interpret the vertex properties in the context of the patents. In Section 4.2.4 we use this approach to analyze the US patent citation network and explore the changes in the kinetics from 1976 to 2000. In Section 4.2.5 we discuss some possible implications of our results.

4.2.2 Patentological background

While a similar approach could be applied to many patent systems, including the very important European and Japanese patent systems, we begin our analysis with the United States patent system for which an extensive database of citations has been made available through the work of Hall et al.

[2003].

An application for a U.S. Patent is filed in the U.S. Patent and Trademark Office (USPTO). A patent examiner at the USPTO determines whether to grant a patent based on a number of criteria, most important of which for present purposes are the requirements of novelty and non-obviousness with respect to existing technology. Once a patent is issued by the USPTO, it is assigned a unique patent identification number. These numbers are sequential in the order in which the patents were granted.

Novelty and nonobviousness are evaluated by comparing the claimed invention to statutorily defined categories of "prior art", consisting in most cases primarily of prior patents. Patents are legally effective only for a limited term (currently twenty years from the date of application), but remain effective as "prior art" indefinitely. Inventors are required to provide citations to known references that are "material" to patentability, but are not required to search for relevant references (though they or their patent attorneys often do so). During consideration of the application, patent examiners search for additional relevant references.

Patent citations include potential prior art that was considered by the examiner. They thus reflect the judgment of patentees, their attorneys, and the USPTO patent examiners as to the prior patents that are most closely related to the invention claimed in an application. Patent citations thus provide, to some approximation, a "map" of the technical relationships between patents in the U.S. patent system. This "map" can be represented by a directed network, where the nodes are the patents and the directed edges the citations. Our research attempts to gain insight from this "map".

The patent database we use for the analysis in this paper was created by Hall, Jaffe and Trajtenberg based on data available from the US Patent Office [Hall et al., 2003]. It is available online at `http://www.nber.org/patents/`. The database contains data from over 6 million patents granted between July 13, 1836 and December 31, 1999 but only reflects the citations made by patents after January 1, 1975: more than 2 million patents and over 16 million citations. Citations made by earlier patents are also available from the Patent Office, but not in an electronic format. The Hall, Jaffe and Trajtenberg database also contains additional data about the included patents, which was described in detail by Hall et al. [2003].

4.2.3 Modeling patent citation networks

The raw citation data gives us a complete history of citations made and received by each patent. Our goal is to determine whether the evolution of the patent network may be consistently described in terms of variables commonly used in understanding the evolution of complex networks and then to extract the time dependence of the network growth from the detailed history. We assume as an initial matter (an assumption which turns out to be consistent with the data) that the evolution of the network may be approximated by a discrete time, discrete space stochastic dynamic system. Time is measured in patent number units, so that each "time step" represents the citations made by a single patent. (Though we often "bin" the data from a range of patent numbers to obtain sufficient statistics for the analysis.) In our model, each patent is described by two variables:

1. k, the number of citations it has received up to the current time step and

2. l, the age of the patent, which is simply the difference between the current time step (as measured in patent numbers) and the patent number. Because a given patent may cite more than one other patent, several citations may be made in one time step.

These two variables define what we call the "attractiveness" of a patent, $A(k, l)$, which determines the likelihood that the patent will be cited when the next citation is made. In every time step the probability that an older patent will be cited is proportional to the older patent's attractiveness multiplied by the number of citations made in that time step. We find that this simple model gives a very good approximation of the observed kinetics of the growth of the patent citation network.

The $A(k, l)$ function determines the evolution of the network. It describes the average citation preferences of the citing patents (the inventors and patent examiners in reality). In this study, we measure and analyze $A(k, l)$ for the United States patent system during the time period covered by our data. We find first that the parameterization by k and l consistently describes the average kinetics of the patent citation network. Of course, underlying patent citations are patentee and patent examiner evaluations of the significance of the cited patent and the technological relationship between the citing and cited patents that our probabilistic approach cannot capture. The way in which these "microscopic dynamics" are translated into the average behavior that we observe remains an open question.

Note that this mathematical framework is an assumption about the evolution of the patent network but it is also a model in the sense that it is a simplified description of the system and that the sensible results we obtain from our numerical procedure confirm that it is a reasonable model. Similar degree and age based models have been studied by others Dorogovtsev and

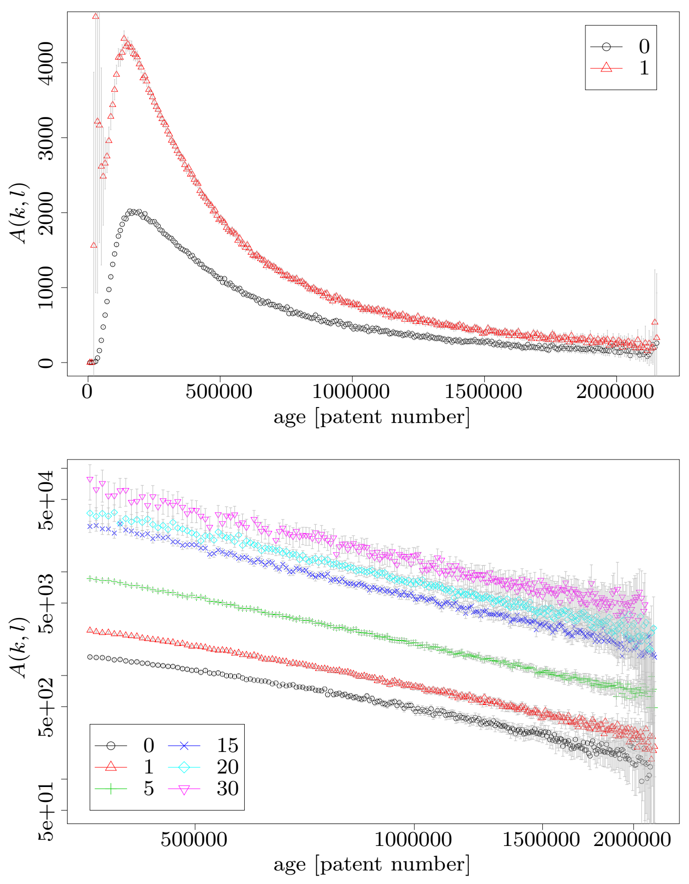


Figure 4.3: The measured attractiveness $A(k, l)$ as a function of age l for various fixed values of in-degree, k. The bottom figure shows only the decreasing tail on log-log scales.

Mendes [2000], Zhu et al. [2003], Klemm and Eguíluz [2002], but the work presented here is different in two respects. First, the degree and age dependence is a general function, we do not assume any particular form; second, we determine the shape of $A(k, l)$ from the patent citation network data.

4.2.4 Results

4.2.4.1 The attractiveness function

The frequentist method described in Section 3.2 was applied to the patent citation network and the forms of $S(t)$ and $A(k, l)$ were determined. Figures 4.3 and 4.4 show sections of the $A(k, l)$ function. For all the figures in this paper we have binned the age values into 300 bins, each containing 7172 patents. Ages and times are measured in patent number units. Figures 4.3 and 4.4 suggest that, for the patent network, the effects of in-degree and age can be separated to a good approximation and that $A(k,l)$ can be written approximately in the form

$$A(k,l) = A_k(k) \cdot A_l(l). \tag{4.3}$$

While this is a reasonable and useful approximation, it is also clear that it is only approximately true. e.g., $A(0,l)$ decays faster than $A(30,l)$, see the second plot in Figure 4.3.

The measured $A_l(l)$ function for the patent citation network has two major features – a peak at approximately 200,000 patent numbers and a slowly decaying tail. (The very large absolute values of $A_l(l)$ are a result of the normalization, $A(0,1) = 1$, and are of no independent significance.) The peak at 200,000 patent numbers corresponds to a large number of what might be called "ordinary", relatively short-term citations. In 1998–1999, 200,000 patent numbers corresponded to about 15 months. The tail is best described by a power-law decay: $A_l(l) \propto l^{-\beta}$ with $\beta \approx 1.6$. The observation of this power law decay is an important result. It indicates that while typical citations are relatively short-term, there are a significant number of citations that occur after very long delays. Very old patents are cited, suggesting that the temporal reach of some innovations, which perhaps can be described roughly as "pioneer", is very long indeed. Moreover, because $A_l(l)$ is approximately independent of k – i.e., approximately the same power law decay is observed even for small k – the power law tail of $A_l(l)$ demonstrates that there is a significant possibility that patents that have gone virtually un-cited for long periods of time will re-emerge to garner citations. This slow power law decay of $A_l(l)$ thus suggests the unpredictability of innovative progress.

The measured $A_k(k)$ function increases monotonically with k, as Figure 4.4 suggests. Higher in-degree always means higher attractiveness. Because the citation probability is proportional to the attractiveness, this means that the well-known preferential attachment, or "rich get richer" effect is at work here – the more citations a patent has received, the more likely it is to receive another. The functional form of $A_k(k)$ is a power law over the entire range of k values. $A_k(k) \propto k^\alpha + a$, where $\alpha \approx 1.2014$ and $a \approx 1.0235$. We estimated these parameters using the smaller values of k, for which we have more data. We then checked the results by comparing with more extensive fits.

Preferential attachment and its variations are well studied, see the reviews by Albert and Barabási [2002] and by Newman [2003a]. Linear preferential attachment ($\alpha = 1$) without aging has been shown to result in a

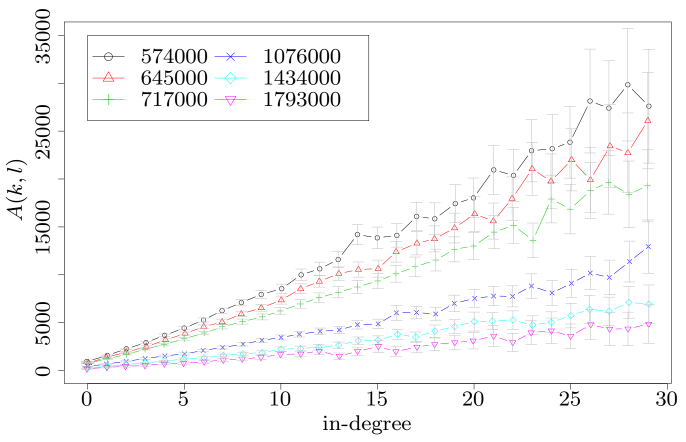


Figure 4.4: The measured attractiveness $A(k,l)$ as a function of in-degree, k, for various fixed values of age, l.

degree distribution (frequency of nodes with degree k) with a power law tail [Albert and Barabási, 2002, Newman, 2003a]. Krapivsky et al. [2000] have studied nonlinear preferential attachment. In the model they studied there was no aging, $A(k,l) = A_k(k) = k^\alpha + a$. For $\alpha > 1$, as is observed in the patent citation network, their calculations predict a condensation of node connectivity, in the sense that with high probability most of the edges are connected to only a small number of nodes. More specifically, in their model, if $(m+1)/m < \alpha < m/(m-1)$ the number of nodes with more than m incoming edges is finite, even in an infinite network. For the patent network $6/5 < \alpha < 5/4$ suggesting that, if there were no aging, the number of patents receiving more than 5 citations would be very small, though those patents would account for a large fraction of all of the citations. Aging complicates this picture, of course, and likely precludes a complete condensation onto a few nodes. However, the fact that the observed preferential attachment is super-linear does indicate a tendency toward what might loosely be called "stratification" – many nodes with very few citations and a few nodes with many citations.

4.2.4.2 The total attractiveness

The total attractiveness function, $S(t)$, (see Fig. 4.5) of the US patent system increases with time. The initial steep increase is only a finite size effect and comes from the fact that the citations made by pre-1975 patents are missing from our database. From about 1984 on, however, $S(t)$ displays a slow but steady increase. One way to interpret this increase is that the probability

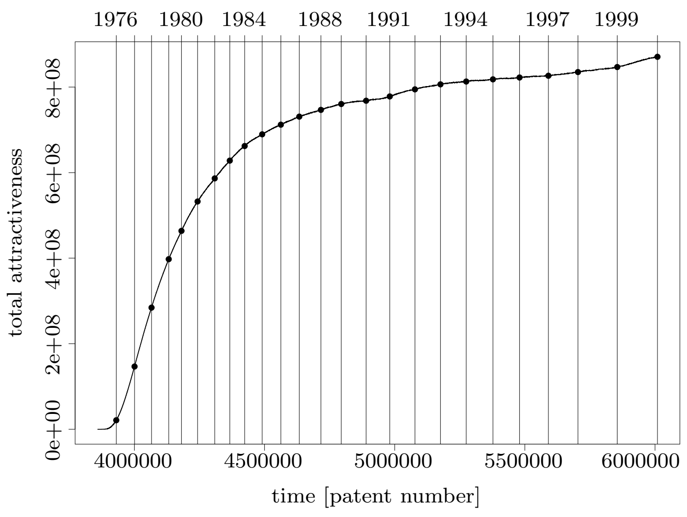


Figure 4.5: The total attractiveness $S(t)$ of the patent network versus time in units of patent numbers. For ease of reference the time in years is indicated by filled circles and vertical lines.

that a patent will be cited by a given citation (which is proportional to $1/S(t)$) is decreasing as the size of the network increases. The decrease is determined in part by the rate at which patents age, which determines the number of patents "available" for citation.

The probability that patent i will be cited in a given time step (in other words, by a particular patent rather than by a particular citation) is

$$P[k_i(t+1) = k_i(t) + 1] = E(t)\frac{A(k_i(t), l_i(t))}{S(t)} \tag{4.4}$$

The average number of citations made by each patent (and hence, since we measure time in units of patents, the number of citations made in each time step, $E(t)$), has increased approximately linearly with time in the real patent citation network, e.g., it was 4.69 in 1975 and 10.66 in 1999. See Fig. 4.6. The probability that a new patent ($k = 0, l = 1$) will be cited by the next patent is thus given by $E(t)/S(t)$, which is shown in Fig. 4.6. From this plot one can see that the increase in the number of citations being made slightly outweighs the increase in $S(t)$, so that the probability that a new patent will be cited has increased over time, despite the increasing $S(t)$. Despite the persistent relevance of some old patents indicated by the power law tail in $A_l(l)$, new patents do not get "lost in the crowd" the way we might have predicted from simple models. Instead, patentees and patent examiners have on average increased the number of citations made by each patent to more

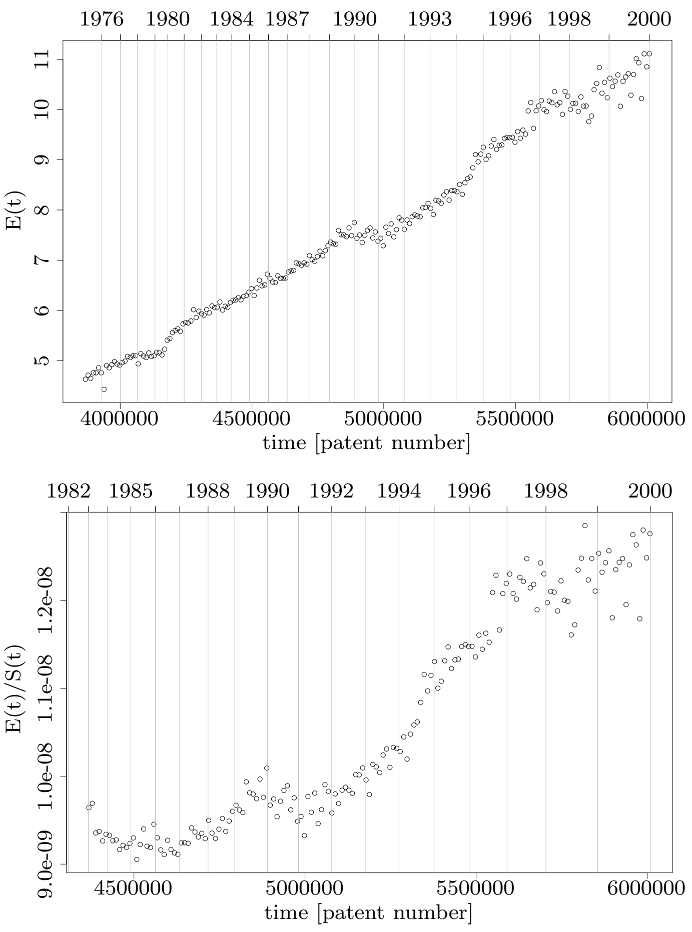


Figure 4.6: The top figure shows the number of citations made per patent, $E(t)$, as a function of time in units of patent number. The bottom figure shows $E(t)/S(t)$, corresponding also to the probability that a new patent with $k = 0$, $l = 1$ will be cited, as a function of time in units of patent number.

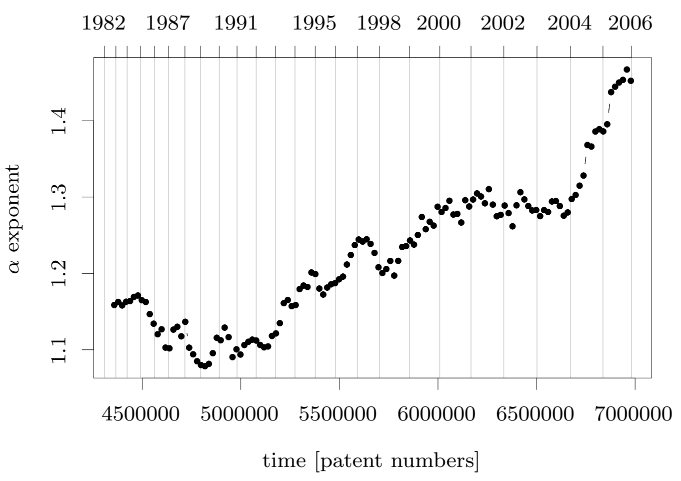


Figure 4.7: The measured value of α as a function of time, measured as described in the text. The time in years is indicated by the grey vertical lines.

than compensate for the increasing $S(t)$.

4.2.4.3 Change in the patent system dynamics

While it is well known that there has been a significant increase in the number of US patents granted per year since 1984 [Jaffe and Lerner, 2004, Hall, 2005], the underlying reason for this increase is widely disputed. Has there simply been an acceleration of technological development in the last twenty years or has there been a more fundamental change in the patent system, perhaps, as many have suggested, as a result of increased leniency in the legal standard for obtaining a patent [Jaffe and Lerner, 2004]? A complete answer to this question is far beyond the scope of the present investigation. However, our model does permit us to ask whether there has been any deep change in the growth kinetics of the patent citation network. Because we measure time in units of patent number, a mere acceleration of technological progress should leave $A(k, l)$ unchanged in patent number "time". A change in $A(k, l)$ indicates some other source of change.

Thus far, we have assumed a time-independent $A(k, l)$, which is reasonably consistent with our observations. In this section, we relax this assumption to ask the more subtle question of whether there has been a change in patent system kinetics over and above the acceleration that is already reflected in our choice of time units. Specifically, we allow α and β to vary with time and ask whether there has been a significant change in these parameters between 1980 and 2000.

To answer this question we measured the parameters of the system as functions of time. To perform the fits, we averaged over a 500,000-patent sliding time window and calculated the parameters after every 100,000 patents. The measured α parameters are plotted in Figure 4.7. There is a significant variation over time. The time dependence of the important β parameter was also explored, but no significant time dependence was observed to within the statistical errors.

The plot for the α parameter shows that there are two regimes. In the first regime, prior to about 1991, α is decreasing slightly with time, while in the second, starting around 1993, there is a significant increase. As noted earlier, the α parameter has some very important consequences for the growth of the network: the higher α, the more "condensed" or "stratified" the network will be. The increasing α in the patent citation network indicates increasing stratification – a smaller and smaller fraction of the patents are receiving a larger and larger fraction of the citations. This change is not simply a result of accelerating numbers of patents being granted, but suggests a more fundamental change in the distribution of patents that are being issued.

We will return to this problem in Section 4.4.2 and examine the change of the patent system dynamics by using other models and better parameter fitting methods.

4.2.5 Conclusions

We have presented a stochastic kinetic model for patent citation networks. Though a complex process underlies each decision by a patent applicant or examiner to cite a particular patent, the average citation behavior takes a surprisingly simple form. The citation probability can be approximated quite well by the ratio of an "attractiveness function", $A(k, l)$, which depends on the in-degree, k, and age in patent numbers, l, of the cited patent, and a time-dependent normalization factor, $S(t)$, which is independent of k and l.

We applied the kernel-based technique to the patent citation network and, though no assumptions were made as to the functional form of $A(k, l)$, the measured $A(k, l)$ function was well described by two approximately separable processes: preferential attachment as a function of in-degree, k, and power law age dependence. The interplay of these two processes, along with a growth in the number of citations made by each patent, governs the emerging structure of the network. Particularly noteworthy are our finding that the preferential attachment is super-linear, implying that patents are highly stratified in "citability", and our finding of a power law tail in the age dependence even for small k, indicating not only that some patents remain important for very long times, but also that even "dormant" patents can re-emerge as important after long delays.

We also used our technique to investigate the time dependence of the growth kinetics of the patent citation network. Overall, we find that the increasing number of patents issued has been matched by increasing citations made by each patent, so that the chance that a new patent will be cited in the next time period has increased over time. This result suggests that on average patents are not becoming less "citable". However, we also find that there has been a change in the underlying growth kinetics since approximately 1993. Since that time, preferential attachment in the patent system has become increasingly strong, indicating that patents are more and more stratified, with fewer and fewer of the patents receiving more and more of the citations. A few very important, perhaps "pioneer", patents seem to dominate the citations. This trend may be consistent with fears of an increasing patent "thicket", in which more and more patents are issued on minor technical advances in any given area. These technically dense patents must be cited by patents that build upon or distinguish them directly, thus requiring that more citations be made, but few of them will be of sufficient significance to merit citation by any but the most closely related patents. These observations are consistent with recent suggestions that patent quality is decreasing as a result of insufficient standards of non-obviousness. See, for discussion of these issues, e.g., Federal Trade Commission [2003], Jaffe and Lerner [2004], Merrill et al. [2004] and references therein.

4.3 The dynamics of scientific collaboration networks

In this section we briefly present the results of applying our methods to a non-decaying network: the cond-mat collaboration network. In this network a vertex is a researcher who published at least one paper in the online arXiv cond-mat archive (see `http://www.arxiv.org`) between 1970 and 1997 (this is the date when the paper was submitted to cond-mat, not the actual publication date, but most of the time these two are almost the same). There is an edge between two researchers/vertices if they've published at least one paper together. The data set contains 23,708 papers, 17,636 authors and 59,894 edges.

First, we measured the attachment kernel for this network based on the degrees of the two potential neighbors. See Fig. 4.8 for the $A_{\text{cond-mat}}(d^*, d^{**})$

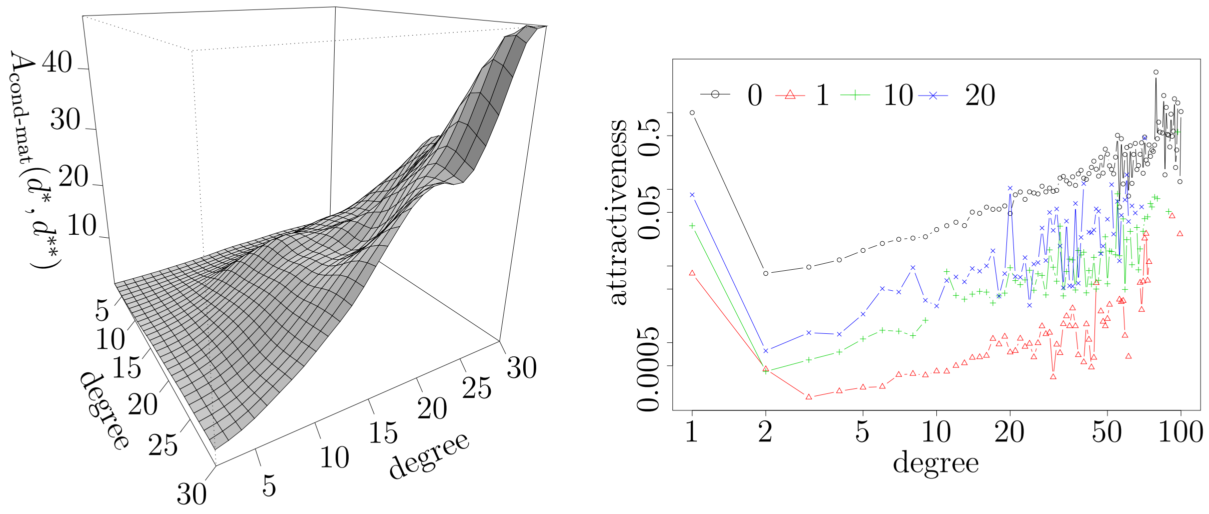


Figure 4.8: The attachment kernel for the cond-mat collaboration network, the surface plot was smoothed by applying a double exponential smoothing kernel to it. The right plot has logarithmic axes. The right plot shows that the kernel function has high values for zero-degree nodes, this might be because a new researcher will usually write a paper with collaborators and thus will have a high probability of gaining new edges immediately.

function. The attractiveness grows with vertex degree, about linearly, except if the degree is zero. Zero degree vertices have very high probability to gain new edges, this is because many papers have contribution from authors, who had no other paper yet in the cond-mat database. These results show that a second, different kernel function might be appropriate for predicting the citations of the newly added authors, Newman [2001b] used a similar framework.

We've tried to fit various functional forms to the two-dimensional attachment kernel function to check which is a better description of the dynamics. See Fig. 4.9 for the shape of the fitted functions and Table 4.1 for the functional forms and the results.

The best fit was obtained by

$$A'_{\text{cond-mat}}(d^*, d^{**}) = c_1 \cdot (d^* d^{**})^{c_2} + c_3 \tag{4.5}$$

where c_i are constants.

Secondly, we used the number of papers as the vertex property, and measured the $A_{\text{cond-mat}}(p^*, p^{**})$ kernel function. We tried to fit the same func-

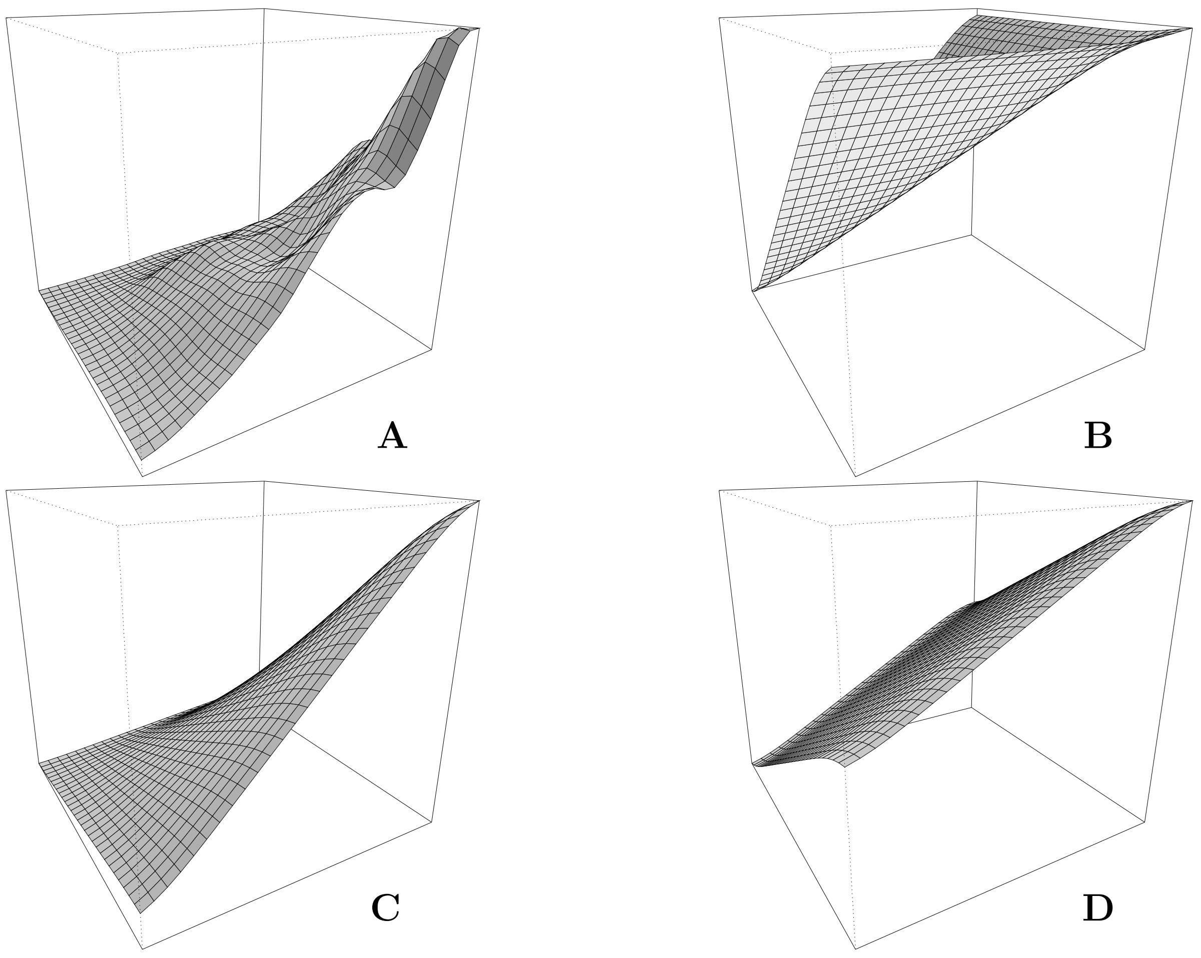


Figure 4.9: **A** shows the smoothed measured degree based kernel function for the collaboration network, **B**, **C** and **D** are fitted functional forms shown in the first three lines of Table 4.1. The best fit is clearly obtained by the multiplicative form.

	Fitted form	Fitted parameters	Fit Error	Fitting method
B	$c_1 \max(d^*, d^{**}) + c_2$	$c_1 = 1.26$, $c_2 = -10.56$	107357.6	Nelder-Mead
C	$c_1 d^* d^{**} + c_2$	$c_1 = 0.0697$, $c_2 = -2.11$	4300.2	Nelder-Mead
D	$c_1(d^* + d^{**}) + c_2$	$c_1 = 1.08$, $c_2 = -18.98$	31348.9	Nelder-Mead
	$c_1 d^* d^{**} + c_2(d^* + d^{**}) +$	$c_1 = 0.0783$, $c_2 = -0.12$,	3532.9	BFGS
	$+c_3 \max(d^*, d^{**}) + c_4$	$c_3 = -0.093$, $c_4 = 1.50$		
	$c_1(d^* d^{**})^{c_2} + c_3$	$c_1 = 0.016$, $c_2 = 1.22$,	3210.4	SANN
		$c_3 = 0.58$		

Table 4.1: Fitting the cond-mat network with an in-degree model. Four optimization methods were run for each functional form to minimize the least square difference: BFGS, Nelder-Mead, CG and SANN, the results of the best fits are included in the table. See [Nocedal and Wright, 1999, Belisle, 1992] for the details of these methods.

	Fitted form	Fitted parameters	Fit Error	Fitting method
B	$c_1 \max(p^*, p^{**}) + c_2$	$c_1 = 0.58$, $c_2 = -2.54$	45930	Nelder-Mead
C	$c_1 p^* p^{**} + c_2$	$c_1 = 0.04$, $c_2 = -0.49$	7470	Nelder-Mead
D	$c_1(p^* + p^{**}) + c_2$	$c_1 = 0.57$, $c_2 = -8.81$	20513	Nelder-Mead
	$c_1(p^* p^{**})^{c_2} + c_3$	$c_1 \ll 1$, $c_2 = 1.59$, $c_3 = 3.14$	5312	SANN

Table 4.2: Fitting the cond-mat network based on the number of papers written by an author. Four optimization methods were run for each functional form to minimize the least square difference: BFGS, Nelder-Mead, CG and SANN, the results of the best fits are included in the table.

tional forms to the measured kernel functions and found that the form

$$A'_{\text{cond-mat}}(p^*, p^{**}) = c_1 \cdot (p^* p^{**})^{c_2} + c_3 \tag{4.6}$$

fits best, although with very different parameters than for the degree based measurement.

The goodness of the degree-based kernel was 2.41, the kernel based on the number of papers has goodness 6.69.

See [Barabási et al., 2002, Newman, 2001a] for other studies on collaboration networks.

4.4 Comparing alternative models

In this section we return to the modeling of the US Patent system. We create alternative models to the in-degree and age based approach, presented in Sec. 4.2, and check whether the change found in Sec. 4.2.4.3 is present in the new models. We compare the expressive power of the various models by calculating their goodness value. The tested models are summarized in Table 4.3.

In-degree In the simplest model the attachment probability depends on the in-degree of the potentially cited vertices only. We found that the kernel

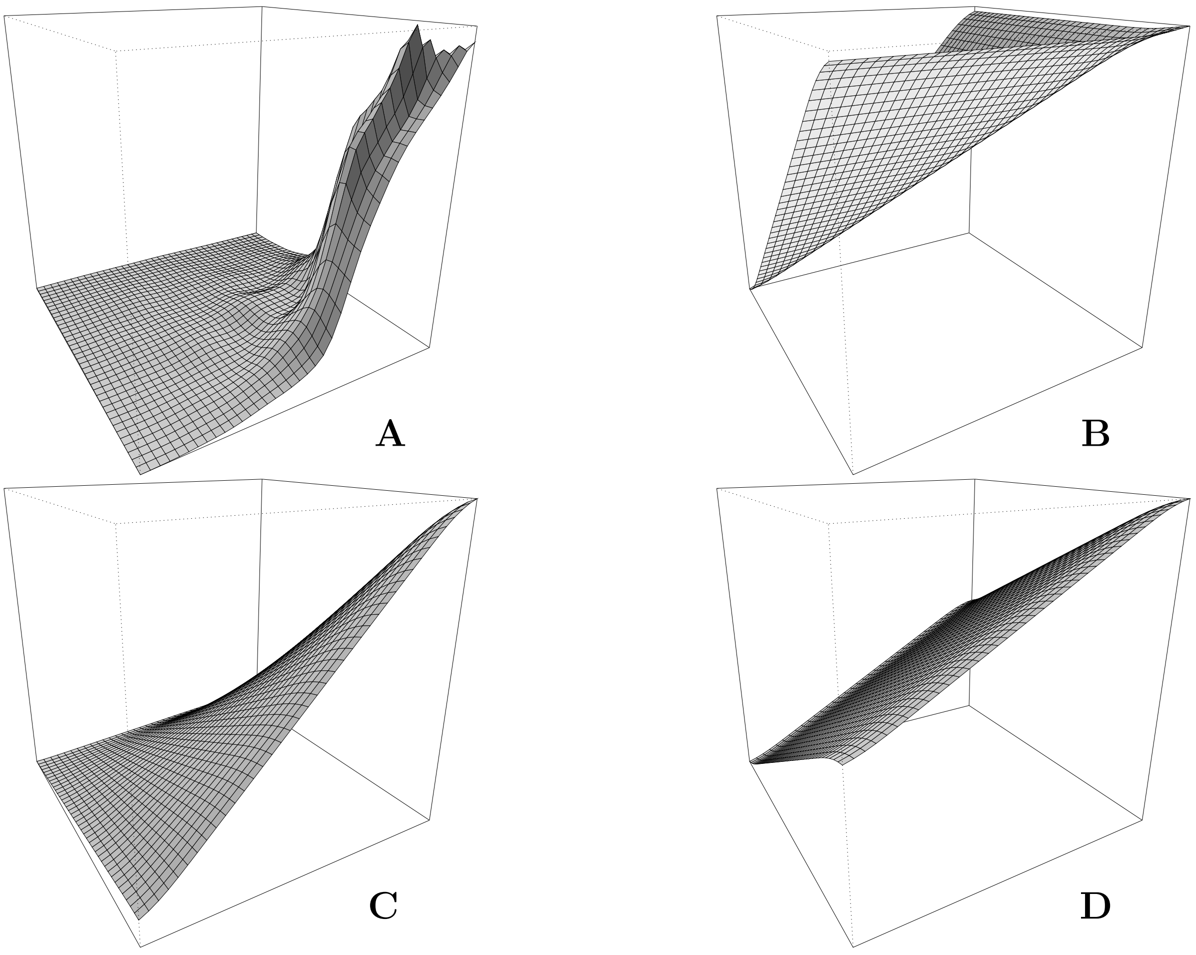


Figure 4.10: **A** shows the smoothed measured based kernel function based on the number of papers, for the collaboration network, **B**, **C** and **D** are fitted functional forms shown in the first three lines of Table 4.2. The best fit is clearly obtained by the multiplicative form.

Properties	*Form*	*Goodness*	*Parameter values*	*Figures*
* in-degree	free	0.3092	–	Fig. 4.11
* in-degree	$d^\alpha + a$	0.3048	$\alpha = 0.99$, $a = 2.62$	Fig. 4.11
in-degree	free	0.2431	–	–
in-degree	$d^\alpha + a$	0.2394	$\alpha = 0.92$, $a = 1.95$	–
i-d, age	free	0.4541	–	Fig. 4.12
i-d, age	double Pareto	0.4373	$\alpha = 1.08$, $a = 1.1$, $\alpha_p = 0.3$, $\beta_p = 2.25$, $t_p = 26.36$	Fig. 4.12
i-d, citing category	free	0.2583	–	Fig. 4.14
i-d, citing category	$d^\alpha + a$	0.2555	See Fig. 4.15	Fig. 4.15
i-d, cited category	free	0.2656	–	Fig. 4.13
i-d, cited category	$c_{\text{cat}}(d^\alpha + a)$	0.2597	$\alpha = 0.89$, $a = 1.95$, $c = [0.92, 1.54, 1.43, 1.06, 0.90, 0.93]$	–
i-d, cited, citing	free	1.1744	–	–
i-d, cited, citing	$c_{\text{cat}}(d^\alpha + a)$	1.1664	See Fig. 4.16	Fig. 4.16
i-d, age, cited, citing	$c_{\text{cat}}\cdot$ double Pareto	1.3454	See Fig. 4.17	Fig. 4.17
forest fire model	free	2.1034	–	–
i-d, forest fire	free	2.2539	–	–

Table 4.3: Summary of the various models fitted to the US patent citation network. Starred models were fitted to network data from 1975 to 2005, the others to data from 1975 to 1999. “i-d” means in-degree, see Eq. 4.7 for the double Pareto form.

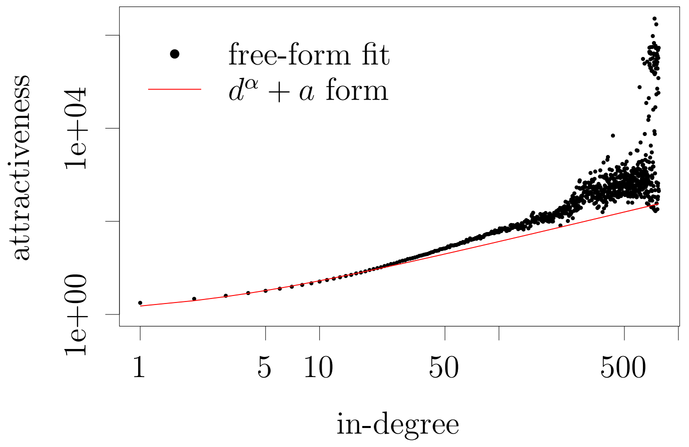


Figure 4.11: The measured *in-degree* based kernel-function for the US patent network. The red line is the maximum likelihood fitted $A(d) = d^\alpha + a$ form. The axes are logarithmic, so $d = 0$ is not included in the plot.

can be well fitted with the form $A(d) = d^\alpha + a$. This form may lead to scale free networks, i.e. networks with power-law in-degree distribution, if the degree dependence is linear, i.e. $\alpha = 1$. Indeed, we found that the in-degree dependent kernel function can be very well fitted with the $A(d) = d^\alpha + a$ form and the value of the α exponent is close to unity.

The fact that the obtained degree-dependent kernel function is a smooth function does not imply that this is the best model of the system and all the citations can be explained simply based on vertex degree. It only tells that this is the best form to come up with *if* we want to model the network based on in-degree only. In other words, if we add additional vertex properties to the model, we might get a better model and each better model can be *averaged out* to the preferential attachment rule.

In-degree and age In addition to the in-degree, we consider the age of the vertices here. By age we do not mean real time, we rather calculate it purely based on the number of vertices. If a vertex was added to the network in time step t_0 and the current time step is t then the age of this vertex is $(t - t_0)/w$, where w is the width of an age-window, typically a couple of thousands of time steps, and the result is rounded to the closest smaller integer. For the patent network we used $w = 7100$.

The in-degree and age dependent model is better than the simple degree-dependent one, which is not a big surprise, an extended model is always at least as good as the original. (Note that practically this is not always

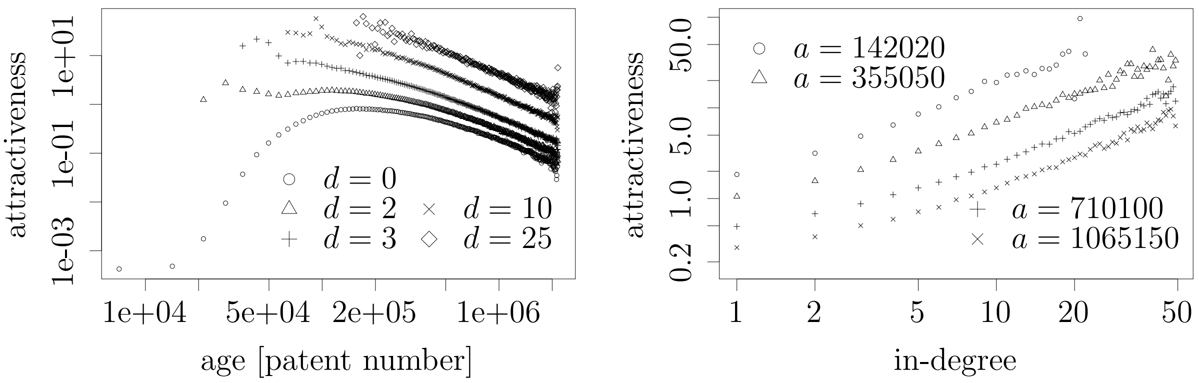


Figure 4.12: Sections from the *in-degree* and *age* based maximum likelihood fitted kernel function for the US patent citation network. Both plots have logarithmic axes.

true, because of numerical errors.) In fact it is about 80% better than the degree-based model, which is significant.

The obtained free-form kernel can be well fitted with a function where the effects of degree and age are separated, $A(d, l) = A(d)A(l)$. The degree-dependent part is the usual preferential attachment form: $A(d) = d^\alpha + a$, of course with a different exponent and different a parameter. We fitted the age-dependent part with a double Pareto function, this is a unimodal function with an initial power-law increase and a power-law decrease in the tail:

$$A(l) = \begin{cases} (l/t_p)^{\beta_p - 1} & \text{if } l \le t_p, \\ (l/t_p)^{-\alpha_p - 1} & \text{if } l > t_p. \end{cases} \qquad (4.7)$$

About 96% of the goodness of the free-form model is preserved with the fitted form. Consequently, the preferential attachment times double Pareto form is a very good model if one wants to model the network based on in-degree and age.

Finding the best kernel function of this form was done by using the BFGS optimization method, without derivatives, as the function has no derivative when $l = t_p$ if $\alpha_p \ne \beta_p$.

4.4.1 Patent categories

Patents are classified into more than 400 patent classes, defined by the US Patent and Trademark Office. Patent classes have further subclasses. Researchers at NBER used these classes to create a patent classification with six

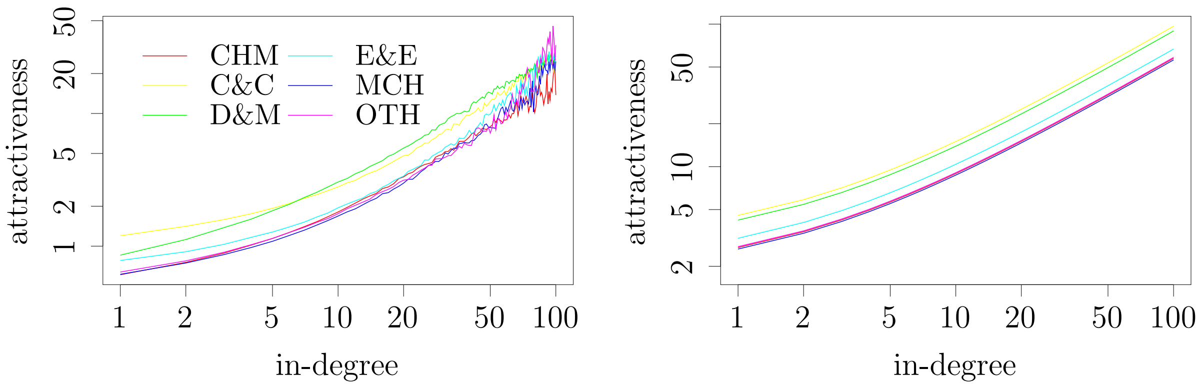


Figure 4.13: The maximum likelihood fitted kernel function based on *in-degree* and *patent category* (of the potentially *cited* vertex). The left plot shows the measured free-form kernel-function. The right plot shows the fitted form $A(d, \text{cat}) = c_{\text{cat}} \cdot (d^\alpha + a)$, c_{cat} is a category-dependent constant. Both plots have logarithmic axes.

big categories and 36 subcategories [Hall et al., 2003]. The six categories are: Chemical, Computers and Communications, Drugs and Medical, Electrical and Electronic, Mechanical and Others.

A natural question is whether the inclusion of patent categories adds a substantial amount to the goodness of the model. We expect a significant goodness increase, based on the fact that the patent network is highly assortative [Newman, 2002, 2003b] with respect to categories, i.e. chemical patents tend to cite chemical patents.

In-degree, cited category First we included the categories in the model by extending the degree-based model with a scalar (1-6) property, the main category of the potentially cited patent. This resulted an about 10% better model. We also fitted a preferential attachment form ($A(d, \text{cat}) = c_{\text{cat}} \cdot d^\alpha + a$) to the network, with the same α and a parameters for each categories, but allowing a different c_{cat} multiplicative constant. This form is justified by Fig. 4.13. This model reveals that the "value" (in terms of the citation preferences) of the six patent categories is approximately the same, there are only slight differences.

In-degree, citing category Then we tried using the categories of the citing patents, this effectively means that we did separate measurements for the categories and obtained six kernel functions. All six kernel functions have

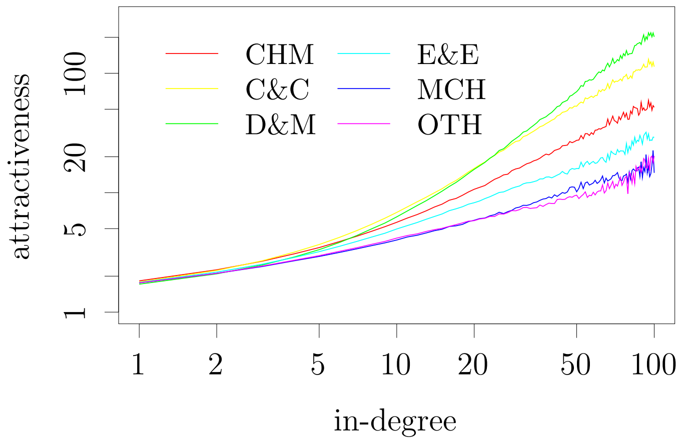


Figure 4.14: Measured kernel-function based on *in-degree* and the *category* of the *citing* vertex. Note that this effectively means that there are separate kernel-functions for the six patent categories. The plot has logarithmic axes.

a similar (preferential attachment kind) shape. This model is only 6% better than the pure degree-based approach.

If we fit the six kernel functions separately using the preferential attachment form $A_c(d) = d^{\alpha_c} + a_c$, then we obtain different α_c and a_c parameters for the six categories, see Fig. 4.15. This shows the preferences (based on in-degree only) of the different category patents when they make their citations. E.g. patents in the 'Mechanical' and 'Others' category tend to "consider" the degree of the cited vertex less than patents in the 'Computers and Communications' and 'Drugs and Medical' category.

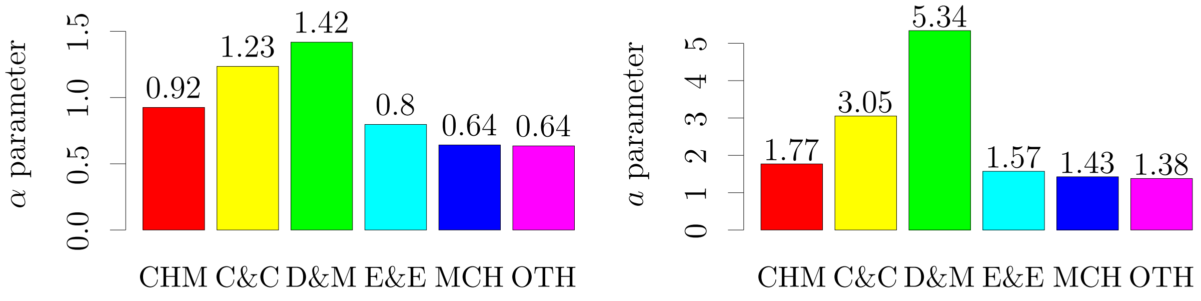


Figure 4.15: Fitted parameters for the *in-degree* and *citing patent category* based model of the form $d^\alpha + a$. Separate kernel functions were fitted for the different patent categories.

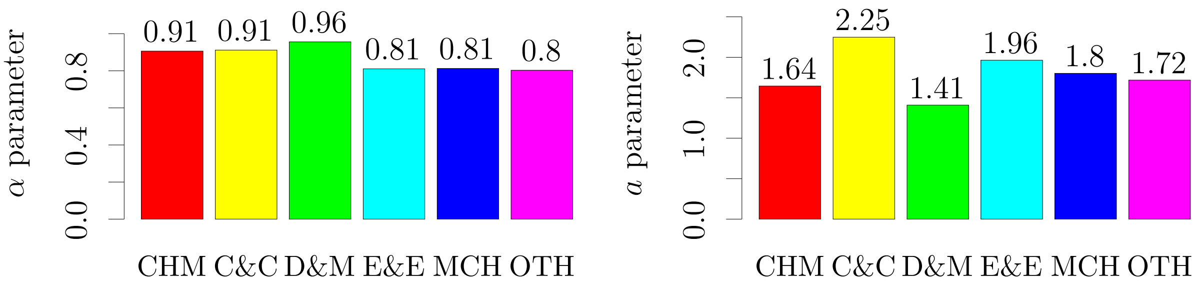


Figure 4.16: Measured parameters based on *in-degree* and the *category* of both the *cited* and *citing* patent. I.e. separate kernel functions of the form $c_{\text{cat}}(d^\alpha + a)$ were fitted for different citing categories.

In-degree, cited category, citing category It is no big surprise that considering the category of both the citing and the cited patents results a much better model. It provides a way to generate a highly assortative network, like ours.

After the free-form maximum likelihood fitting we also tried the formula $A(d, c) = c_{\text{cat}} \cdot (d^\alpha + a)$, where c_{cat} is a category dependent constant, for every *citing* category. Again, we obtained six kernel functions for the six main categories. In this model there are less differences between the categories with respect to the α exponents.

In-degree, age, cited category, citing category Finally, we added the age of the patents to the property vectors and fitted the $A(d, l, c) = c_{\text{cat}} \cdot A(d)A(l)$ form, where $A(d)$ has a preferential attachment form and $A(l)$ is the double Pareto function, as in Eq. 4.7.

4.4.2 Change in the dynamics

Now we return to the question addressed in Sec. 4.2.4.3: did the dynamics of the patent network change around 1990 (supposedly) because of the introduced legal changes? Using the patent categories as vertex properties we can also check whether the change can be found in the "behavior" of all patent categories, or just a few of them are affected.

First we use a model based on in-degree only, this is appropriate, as we're interested in the change of the preferential attachment exponent. Indeed, this model confirms the previous results, and shows that the initially decreasing exponent started to increase around 1990, see Fig 4.18.

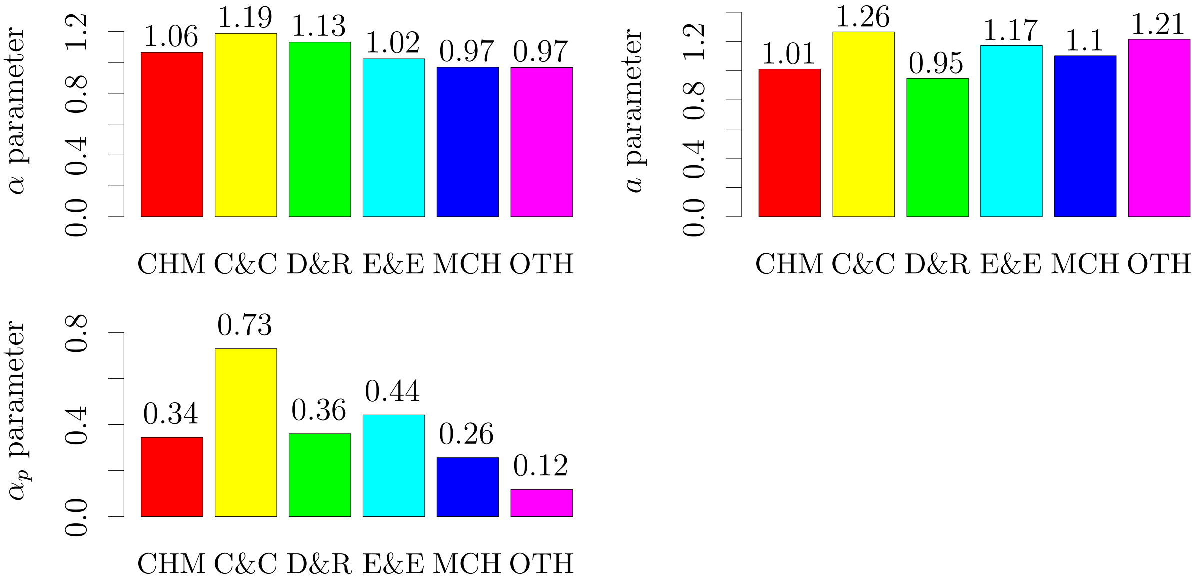


Figure 4.17: Measured parameters based on *in-degree*, *age*, *category* of both the *cited* and *citing* patent. I.e. separate kernel functions were fitted for different citing categories. The fitted form was the product of a degree-dependent term, an age-dependent double Pareto term and a (cited) category-dependent constant. See the text for details.

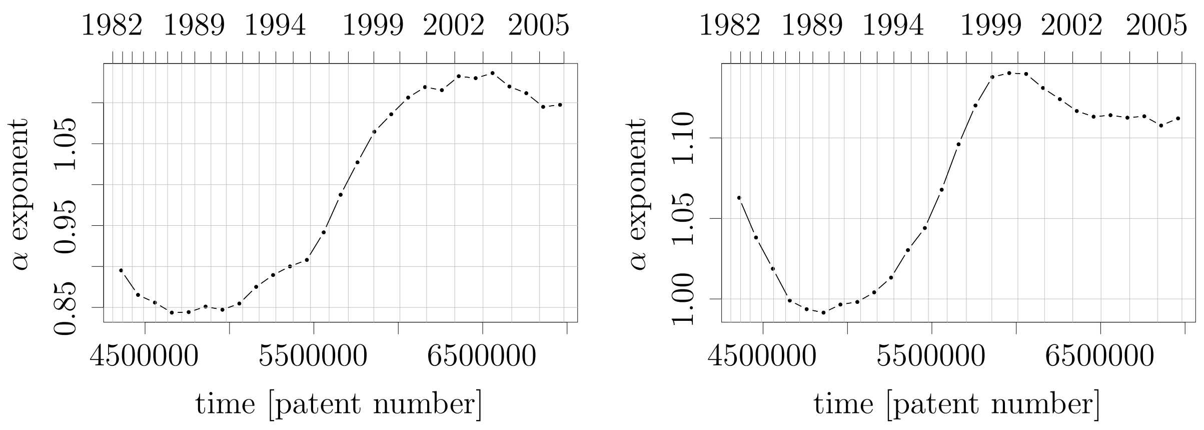


Figure 4.18: Change of the α exponent in the US patent network using two models. The left plot is the maximum likelihood fit of the in-degree dependent model $A(d) = d^\alpha + a$. The right plot is based on an in-degree and age dependent model, the form fitted has an in-degree dependent increasing term and an age dependent unimodal double Pareto term. A sliding time window is used to handle the time-dependence of the parameters, the width of the time window was 500,000 patents and the difference between consecutive windows is 100,000 patents.

Next we repeated the study with a model based on in-degree and age, just like in Sec. 4.2.4.3. This time however we used the maximum likelihood fitting method to fit the parameters of a predefined shape: $A(d, l) = (d^\alpha + a)A(l)$, where $A(l)$ is a double Pareto form. The results, shown in Fig. 4.18 confirm the previous findings, the maximum likelihood method generates much smoother functions than the ad-hoc fitting of the free-form frequentist method.

Next, the category of the citing vertices is used to extract six kernel functions for the six categories. See the results in Fig. 4.19. This model shows that the change is dominated by two patent categories, 'Computers and Communications' and 'Drugs and Medical' and the significant decrease of the exponent for the 'Others' category also stopped suddenly.

We know from Table 4.3 that using the category of both the citing and cited patents boosts the goodness of the kernel a lot. Interestingly enough, in this model the change can be observed for all patent categories, most significantly for the 'Others' category. See the results in Fig. 4.20.

Finally, we added the age of the vertices to the property vectors and the predefined shape $A(d, l, \text{cat}) = c_{\text{cat}} \cdot (d^\alpha + a)A(l)$ was used, where $A(l)$ is the

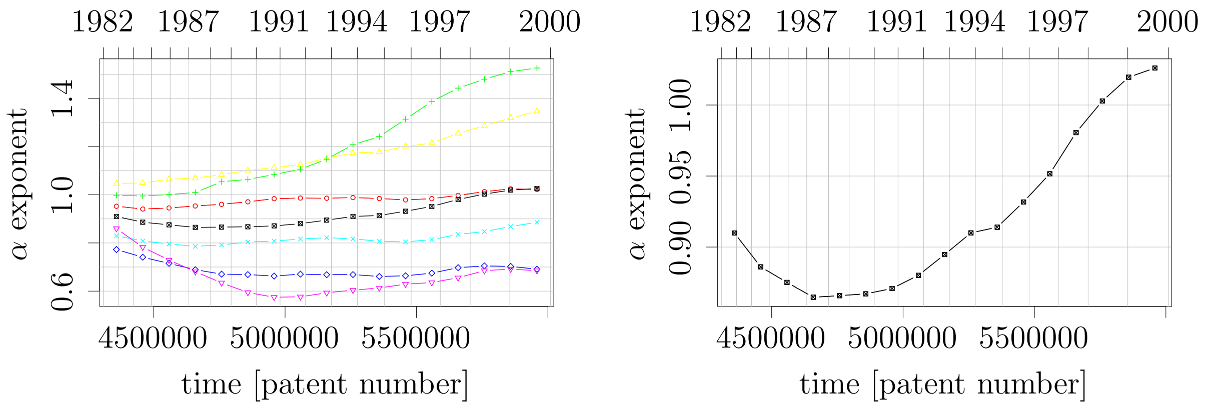


Figure 4.19: Change of the α exponent in the US patent network using a *citing* category and in-degree based model. A separate $A(d) = d^\alpha + a$ model is fitted for each citing category. The "mean" line is calculated by simply averaging the α values of the six categories. The right plot shows only the mean of the six categories. The sliding time window has 500,000 patents and it was applied after every 100,000 patents. The color code is the same as in the previous figures.

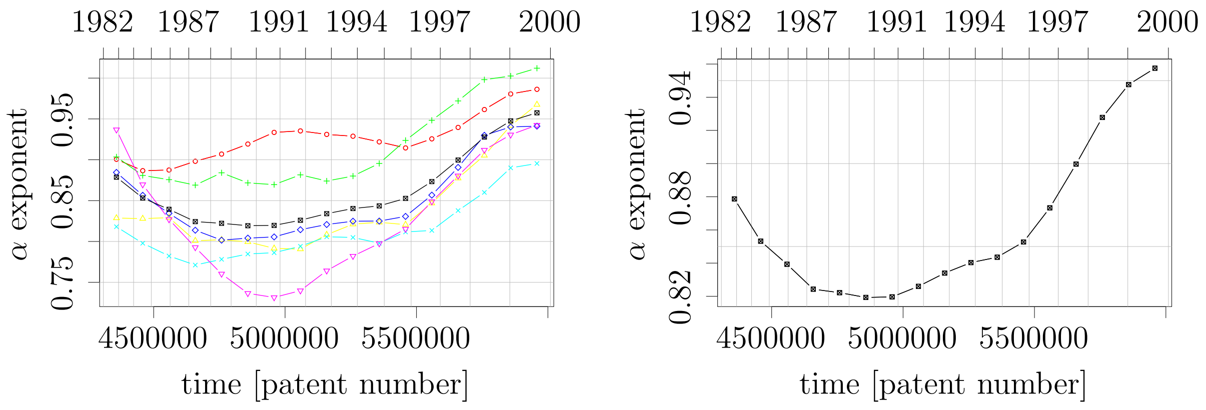


Figure 4.20: Change of the α exponent in the US patent network using an in-degree, cited category and citing category based model. A separate $A(d, \text{cat}) = c_{\text{cat}}(d^\alpha + a)$ model is fitted for each citing category. The "mean" line is calculated by averaging the α values of the six categories. The right plot shows only the mean line. The width and step size of the sliding time window were 500,000 and 100,000 patents respectively.

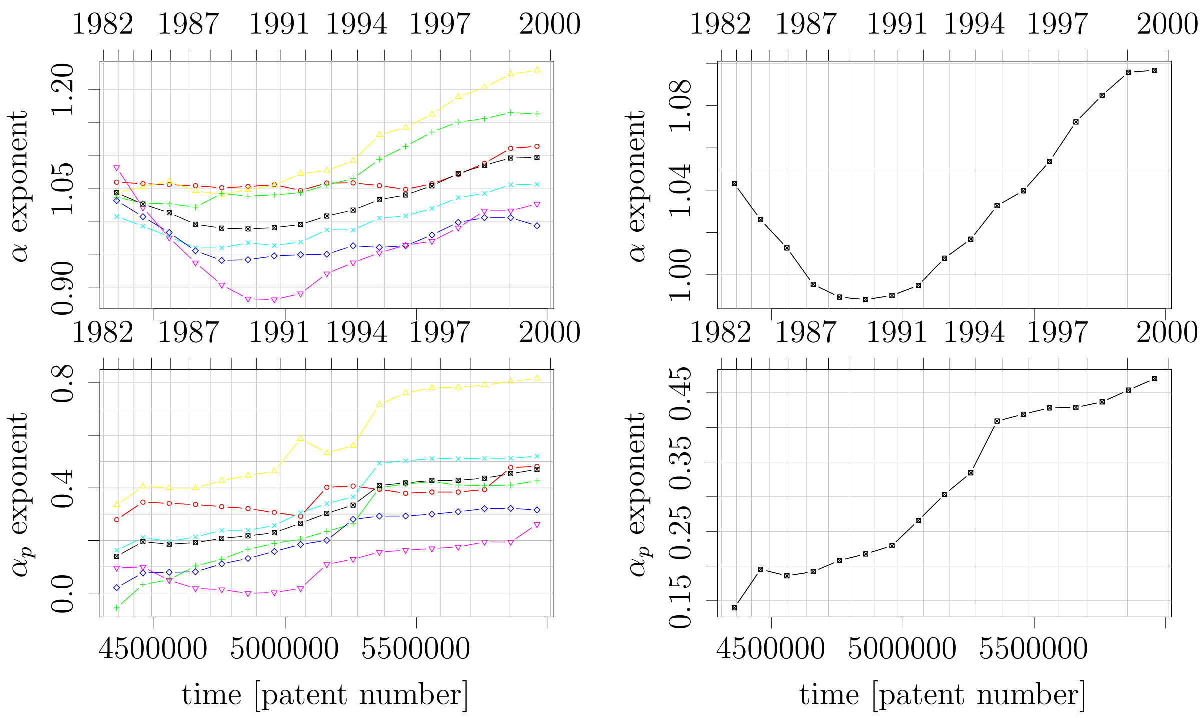


Figure 4.21: Change of the α exponent in the US patent network using an in-degree, age, cited category and citing category based model. A separate model is fitted for each citing category. The model has a degree-dependent term, an age dependent double Pareto term and a (cited) category dependent constant. The width and step size of the sliding time window were 500,000 and 100,000 patents respectively.

double Pareto form from Eq. 4.7. See the results in Fig. 4.21. While the change is clearly present in the α exponent, the α_d exponent of the double Pareto aging function is increasing during the whole period. An increasing α_d parameter means that the patents increasingly prefer to cite younger patents.

To summarize, our previous findings about the change of the α preferential attachment exponent were confirmed in all models we've tried for the patent network. The reason for this change is unclear yet. Although patent law change is one likely cause there might be other explanations, like the recent development in the searching techniques: the patentees and patent examiners started to use electronic search engines when making the citations. If the latter explanation is true, then the change should be seen in other citation networks as well.

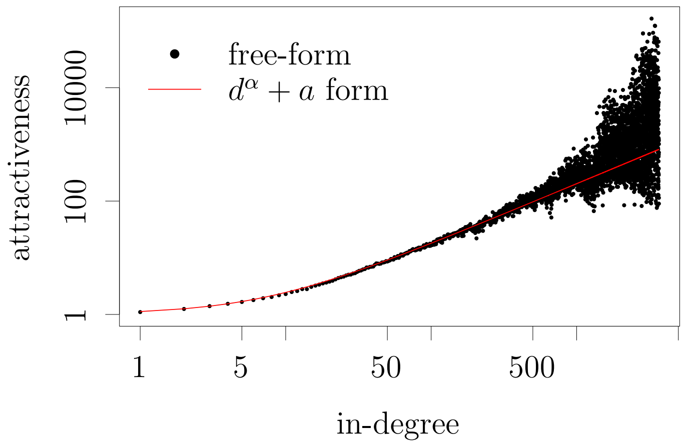


Figure 4.22: *In-degree* dependent model for the APS network. The black dots show the free-form maximum likelihood kernel-function, the red line is the fitted $A(d) = d^\alpha + a$ form.

4.4.3 Comparison to a scientific citation network

In this section we study the citation network of all journals published by the American Physical Society. The network contains 378,077 vertices and 3,615,892 edges. See [Redner, 2005] for an extensive study of the APS network.

First we fitted the same in-degree and in-degree plus age dependent models to the APS network as we did to the patents. The results are similar, although not completely the same. The degree dependent model was approximated with $A_1(d) = d^\alpha + a$, $\alpha = 1.07$, $a = 8.19$. The second, age-dependent model was fitted with $A_2(d,l) = A_d(d)A_l(l) = (d^\alpha + a)l^{-\beta}$ and $\alpha = 1.12$, $a = 0.59$, $\beta = 1.60$ was found. The age dependent $A_l(l)$ kernel interestingly does not have an initial increasing part, there is no need for the double Pareto distribution here. See Figs. 4.22 and 4.23.

Next, we investigated the time-dependence of the α and β parameters. We used a sliding time window, just like for the patent network. See Figs. 4.24 and 4.25 for the results. Although the α values show some variance, the change observed in the patent network can be hardly found here. There is a small increase starting in 1990, but it is just temporal and also much smaller than in the patent network. This is evident by comparing the two networks, see Fig. 4.26.

Based on these results it is not likely that the change found in the patent network dynamics is caused by the sudden improvements in the search technologies.

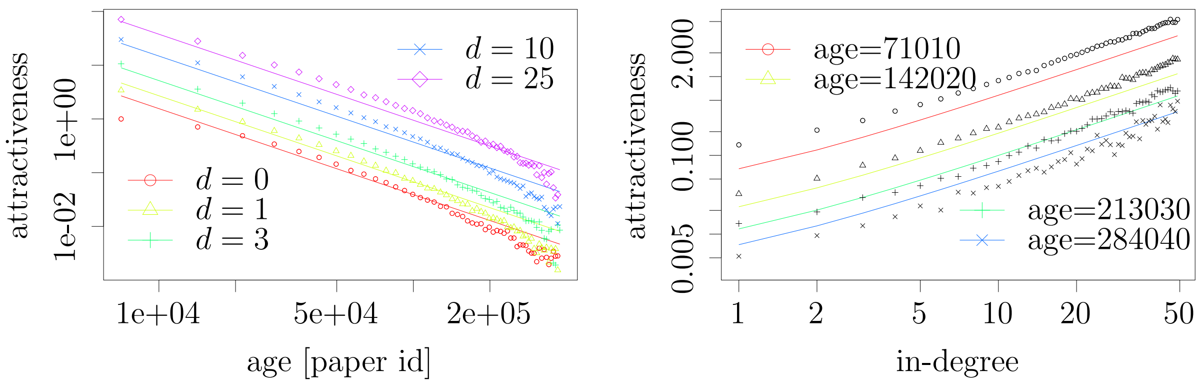


Figure 4.23: Sections from the *in-degree* and *age* dependent kernel function fitted to the APS network. The symbols show the free-form maximum likelihood kernel function, the lines show the best kernel-function of the form $A(d, k) = (d^\alpha + a)k^{-\beta}$. The left plot shows the age-dependence, contrary to the patent network, this has no increasing initial part but similarly has a power-law decay. The degree-dependent right plot shows preferential attachment.

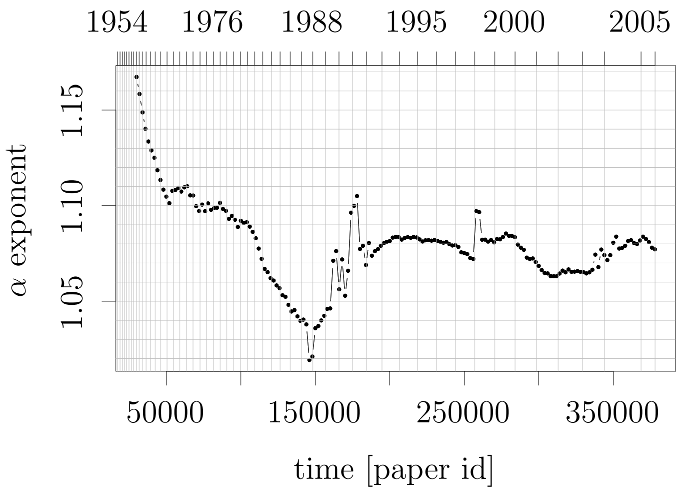


Figure 4.24: Change of the α exponent in the APS network. The plots are based on an *in-degree* dependent model of the form $A(d) = d^\alpha + a$. The sliding time window has the width of 30,000 papers and it was applied after every 2,000 papers.

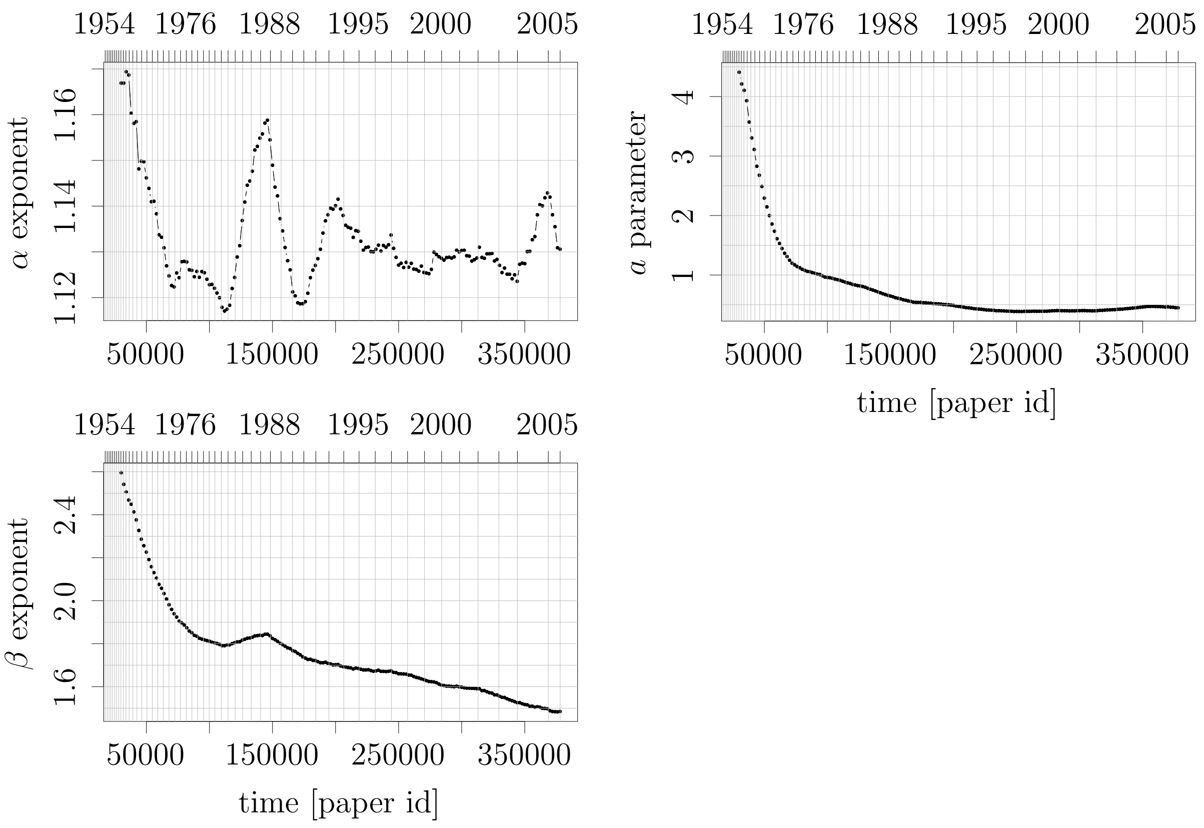


Figure 4.25: Change of the model parameters in the APS network, the model properties were the *in-degree* and the *age* of the vertices. The fitted form was $A(d, k) = (d^{\alpha} + a)k^{-\beta}$. The sliding time window has the width of 30,000 papers and it was applied after every 2,000 papers.

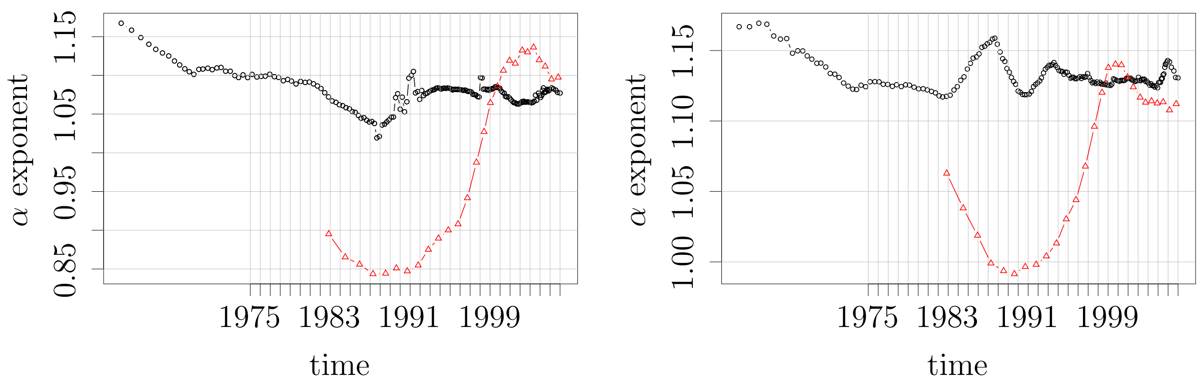


Figure 4.26: Comparing the change of the model parameters in the patent network and the APS network. The data is the same as in Figures 4.18 and 4.24, but the measurement points were shifted horizontally to have a common time axis for the two networks. The left plot is the degree based model, the right is the degree plus age based model.

4.5 Validating network models

4.5.1 The forest fire model

The forest fire model was designed by Leskovec et al. [2005], to explain three facts about real networks:

1. The in-degree and out-degree distributions are power-laws.
2. The networks are densifying in time: the number of edges grows faster than the number of vertices in the graph.
3. The diameter [Bollobás and Riordan, 2004, Albert et al., 1999] of the network is shrinking, it gets smaller and smaller as the network evolves.

Let us briefly explain how the forest fire model works. Let us assume that a new vertex is added to the citation network, it needs to select some other vertices to cite. First it cites a single vertex (more than one in some variations) and then it checks the incoming and outgoing links of this single vertex and with some probability cite these too. Then the process is repeated for the newly cited vertices as well.

The model imitates how a researcher (or inventor or patent examiner) finds relevant papers/patents based on the citation list and the "cited-by" list of the already found relevant papers/patents.

We wanted to validate the forest-fire model for the US patent citation network. This is, however, not straightforward, even if the forest fire model fits into the kernel-based framework. The problem is that the actual order of the citations made by a given vertex is missing from our database: it is naturally not recorded which citations are made first, second, etc.

Neighbors First we created a very simple model with two vertex types: neighbors and non-neighbors. When a new edge is being added to the network all vertices which are in the same components as the citing vertex are considered as neighbors, the others non-neighbors. As there can't be neighbors when the very first citation of a vertex is being made, we omit the first (outgoing) citation of every vertex. The kernel function in this model has just two values, one for the neighbors, one for the non-neighbors. If the value for the neighbors is much higher than the one for the non-neighbors, that indicates the validity of the forest-fire model.

When trying to estimate the kernel for the forest fire model, it does matter in which order the citations of a particular vertex are made. E.g. if we do the measurement with the increasing order of citations (i.e. oldest vertex is cited first, etc.), that favors the "neighbors" vertex type and the kernel function is distorted. If we use the opposite ordering that favors the non-neighbor vertices. By generating synthetic networks we found that by using a random ordering of the citations, the kernel is not biased and on average the correct result is measured.

Indeed, if we fix the kernel function value of the non-neighbors to 1, then the value for the neighbors in the patent network is 12,723.82. A neighbor of an already cited vertex has ten thousand times more probability to get cited (by the same vertex) than a non-neighbor vertex. The goodness of this model is 2.1034, even if we make the first citation of each vertex totally randomly!

Neighbors, in-degree If we add the in-degree of the vertices to the property vector, we get an even better model, one with goodness 2.2539. In this model the first citation is not done randomly, but based on the in-degree of the vertices and then the subsequent citations depend on in-degree and on whether the possibly cited vertex is in the same component as the citing vertex or not. The two kernel functions are plotted in Fig. 4.27.

The forest fire model provides a very good description of the US patent network.

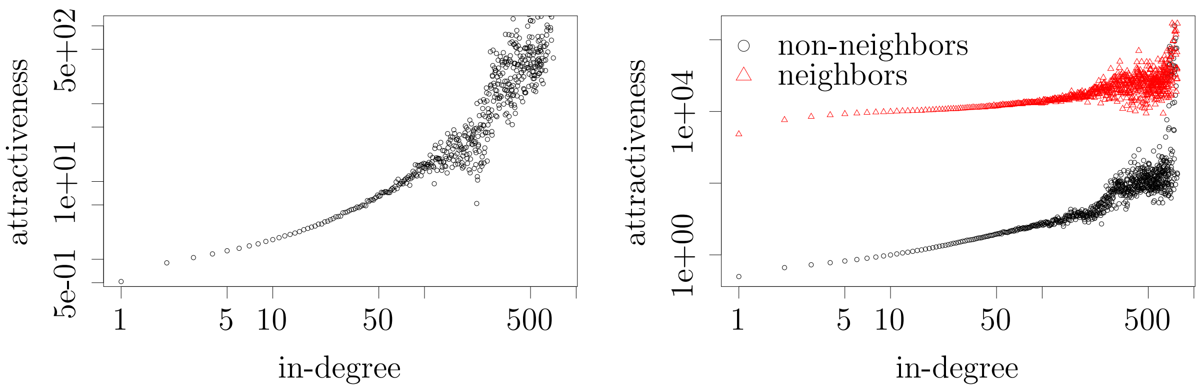


Figure 4.27: The US patent kernel function, forest fire model extended with in-degree. The left plot shows the in-degree based kernel function, which is in effect for the first (outgoing) citation of all vertices. The right plot shows the kernel function for the subsequent citations. It is obvious that neighbor vertices have much higher probability to get cited. Also, note that the kernel for the non-neighbors is much steeper, degree matters more for non-neighbors than for neighbors. Both plots have logarithmic axes.

4.5.2 Preferential attachment is required

Many real world networks feature a power-law degree distribution, these networks are also called ‘scale-free’, as the power-law distribution is invariant to the (multiplicative) scaling of the independent variable.

First Price [1976] showed that the preferential attachment mechanism [Barabási and Albert, 1999] is capable of generating scale-free networks. This mechanism corresponds to an in-degree dependent kernel-function of the form: $A(d) = d + a$, where the (small) a increment ensures that zero-degree vertices have non-zero attractiveness. This model always generates a scale-free network (with respect to the in-degree distribution) with exponent $\alpha = 3$, independently of the out-degree distribution:

$$P[d = k] = (k + b)^{-\alpha}. \tag{4.8}$$

Here we will show that the preferential attachment is not only sufficient to create scale-free networks, but it is also required.

We state the following. If a network has a stationary scale-free in-degree distribution, and we take the limit of the infinite network, then the best in-degree based kernel function describing the evolution of the network is of the form

$$A(d) = d + a \tag{4.9}$$

with some $a > 0$ parameter.

If the network is scale-free and stationary, then

$$p_i = (i + a)^{-\alpha}, \qquad (i \geq 0) \tag{4.10}$$

holds, p_i is the ratio of vertices with in-degree i, and $a > 0$. This implies

$$\frac{M_i(t)}{M(t)} \propto \sum_{j=i+1}^{\infty} p_j = \sum_{j=i+1}^{\infty} (j + a)^{-\alpha}, \qquad t \to \infty, \tag{4.11}$$

where $M_i(t)$ is the number of edges citing i-degree vertices up to time step t and $M(t)$ is the total number of edges up to time step t.

If the network is stationary then the best in-degree based kernel function can be given as (see Sec. 3.3.1.4)

$$A(0) = \frac{M_0}{p_0}, \quad A(1) = \frac{M_1}{p_1}, \quad \ldots \quad A(i) = \frac{M_i}{p_i}, \quad \ldots, \tag{4.12}$$

where

$$M_i := \lim_{t \to \infty} \frac{M_i(t)}{M(t)}. \tag{4.13}$$

In our case this means

$$A(i) = (i+a)^{\alpha} \sum_{j=i+1}^{\infty} (j+a)^{-\alpha} \propto (i+a)^{\alpha}(i+a)^{-\alpha+1} = i+a, \qquad (i \geq 0), \tag{4.14}$$

where we used

$$(i + a)^{-\alpha+1} \propto \sum_{j=i+1}^{\infty} (j + a)^{-\alpha}. \tag{4.15}$$

Note that the "proof" above is independent of the value of α. This means that *if* the assumptions are valid then the best description is *linear* preferential attachment, independently of the exponent. Since it is quite unlikely that this would be true (actually one can check with a couple of simulated networks that it is not), the assumptions must not be valid for every possible α exponent. In other words, the proof hints that the assumptions can be valid only if $\alpha = 3$, in this case we know that there is an in-degree based kernel function which is able to generate the network. This means that scale-free networks with exponents different than $\alpha = 3$ *cannot* be generated with *any* in-degree based kernel-function. In other words, an in-degree based kernel never generates a scale-free network, except if $A(d) = d + a$, in which case it generates one with exponent $\alpha = 3$.

These "hints" are in good agreement with the fact that non-linear preferential attachment ($A(d) = d^\beta + a$, $\beta \neq 1$) never leads to a scale-free network. If we assume that a kernel function with high goodness is able to reproduce the degree distribution, then our proof extends this to other kernel-functions: no in-degree based kernel function leads to a scale-free network, except in the case of linear preferential attachment.

Naturally, we do not state that there are no other mechanisms capable of generating scale-free degree-distributions [Dorogovtsev et al., 2000, Dorogovtsev and Mendes, 2001, Dorogovtsev et al., 2001b,c, Tadic, 2001, Ergun and Rodgers, 2002, Dorogovtsev and Mendes, 2000, Vazquez, 2001], nor that in a scale-free network the choices of the participating actors are based on the degree of the vertices only. But we state that whatever mechanism generated a scale-free network, when we do an in-degree kernel based measurement on it, we will get linear preferential attachment as the result.

A similar statement can be proved for graphs with exponential (in-)degree distribution:

If the network has a stationary exponential in-degree distribution, and we take the limit of the infinite network, then the best in-degree based kernel function describing the evolution of the network is of the form

$$A(d) = 1. \tag{4.16}$$

We follow the same way as in the scale-free case. If the exponential degree-distribution is stationary, then

$$p_i = e^{-i}, \qquad (i \geq 0) \tag{4.17}$$

holds, p_i is the ratio of vertices with in-degree i. This implies

$$\frac{M_i(t)}{M(t)} \propto \sum_{j=i+1}^{\infty} p_j = \sum_{j=i+1}^{\infty} e^{-j}, \qquad t \to \infty, \tag{4.18}$$

where $M_i(t)$ is the number of citations to i-degree vertices up to time step t and $M(t)$ is the total number of edges up to time step t.

If the network is stationary then the best in-degree based kernel function can be given as

$$A(0) = \frac{M_0}{p_0}, \quad A(1) = \frac{M_1}{p_1}, \quad \dots \quad A(i) = \frac{M_i}{p_i}, \quad \dots, \tag{4.19}$$

where

$$M_i := \lim_{t\to\infty} \frac{M_i(t)}{M(t)}. \tag{4.20}$$

In our case this means

$$A(i) = e^{i} \sum_{j=i+1}^{\infty} e^{-j} \propto e^{i} e^{-i} = 1, \qquad (i \geq 0), \tag{4.21}$$

where we used

$$e^{-i} \propto \sum_{j=i+1}^{\infty} e^{-j}. \tag{4.22}$$

5

Other methods for the inverse problem

THIS CHAPTER LOOKS over all other methods I could find for solving inverse problems of network dynamics, deducing the dynamical (actually kinetic) equations of the graph evolution.

Normally this part of a dissertation goes *before* the actual presentation of the work, that would, however, falsely suggest that it has influence on our work, which is—unfortunately—not true.

5.1 Ad-hoc methods

We call the methods of this section ad-hoc, because while they were developed for solving inverse problems, the important generalization to use arbitrary vertex or network properties as the driving force, was not considered.

Barabási et al. [2002] analyzed collaboration networks in mathematics and biology. Their database spans over eight years, and one year is considered as the time step. Δk, the number of new links was counted for every vertex degree k, accumulated in the years before the measurement, but considering only the edges from the new authors, those who had not collaborated with anyone in the previous years.

Another kernel function was used to describe the formation of connections between "old" authors, who already had other collaborators, so-called internal connections. This is a symmetric, degree based kernel $A(d_1, d_2)$, and the normalization factor was simply the number of pairs of vertices with d_1 and d_2 degrees. The method roughly corresponds to one step of the frequentist iteration discussed in Section 3.2.

Both kernel functions show approximately linear preferential attachment.

A similar framework, essentially kernel functions based on degree, were used by Palla et al. [2004].

Newman [2001b] specifically aimed to find preferential attachment in two collaboration networks, one in physics, one in biology. He used degree-dependent symmetric kernel functions and found preferential attachment with exponents close to one. His method is basically the same as our frequentist method (Sec. 3.2), with always using the number of vertices as the normalizing factor instead of $S(t)$. In other words his solution is—almost—the same as doing one step in our frequentist iteration. The 'almost' is because he considers the two ends of an edge joining to two vertices independently, so this is not exactly an $A(d_1, d_2)$ kernel function but something close to it. We know from our numeric results that one iteration of the frequentist method is not enough, it leads to false results, except when the $S(t)$ total attractiveness is indeed proportional to the number of vertices in the network.

Jeong et al. [2003] measure preferential attachment in various networks. They assume a kernel function with shape $A(d) = d^\alpha$ and estimate it based on the number of citations gained in a ΔT time interval. They claim that by using short ΔT time intervals they are able to eliminate the time dependence of the $S(t)$ normalization factor. While this seems plausible, it is unclear how small these time steps should be and how much error we still introduce with the non-proper normalization. Each analyzed network showed closely linear preferential attachment.

Valverde et al. [2007] give an inverse approach, based not solemnly on degree like all the works discussed so far, but also on the age of the vertices. Just like Jeong et al. [2003] they use short time steps to eliminate the effect of the time-dependent normalization factor and find preferential attachment, while fitting the age-dependent component with a Weibull distribution. This is consistent with other studies [Börner et al., 2004].

5.2 Exponential random graphs

The theory of exponential random graphs is without question the most well known, best elaborated and researched method for inverse problems in graphs. There are several ways to introduce this model, we will use one which is close to the spirit of this dissertation and will roughly follow a lecture by Hunter [2006].

The exponential random graph model defines a probability distribution over all graphs with a fixed number of vertices, n. The probability distribution has the form

$$P_\theta(X = x) \propto \exp\{\theta^T s(x)\}, \tag{5.1}$$

with the normalization factor $c(\theta)$ this is

$$P_\theta(X = x) = \frac{\exp\{\theta^T s(x)\}}{c(\theta)}, \tag{5.2}$$

where X is a random variable, a random network on n vertices, $s(x)$ is a vector of graph properties measured on x, e.g. the number of edges, the number of vertices with degree three, the transitivity of the network, the number of connections between men and women in a social network, etc. These will serve as explanatory variables. θ is a vector of coefficients, they should be extracted from the data, in a way to fit the observed network(s) best.

The problem, however, is the normalization factor $c(\theta)$, as it should be summed over all the possible graphs with n vertices, in the most general case without taking into account graph isomorphism:

$$c(\theta) = \sum_{\substack{\text{all } y \text{ graphs} \\ \text{with } n \text{ vertices}}} \exp\{\theta^T s(y)\}. \tag{5.3}$$

Since replacing $s(x)$ with $s(x) - s(x^*)$, where x^* is a given network, leaves the $P_\theta(X = x)$ probabilities unchanged, we can “recenter” $s(x)$ to have $s^*(x^*) = 0$. In the following we assume that $s(x^*) = 0$, 0 is a vector here.

We want to fit θ by maximizing $P_\theta(X = x^*)$, the probability of the observed network should be the highest possible. If $s(x^*) = 0$ then $\exp\{\theta^T s(x^*)\} = 1$ holds and the log-likelihood function is

$$\mathcal{L}(\theta) = -\log c(\theta) = -\log \sum_{\substack{\text{all } y \text{ graphs} \\ \text{with } n \text{ vertices}}} \exp\{\theta^T s(y)\}. \tag{5.4}$$

Obviously, $\mathcal{L}$ cannot be calculated directly, as the number of graphs with n vertices grows exponentially with n and is very high even for small n. We use another approach, based on Markov Chain Monte Carlo methods.

First we observe that if θ_0 is fixed then

$$E_{\theta_0}[\exp\{(\theta-\theta_0)^T s(X)\}] = \frac{c(\theta)}{c(\theta_0)}, \tag{5.5}$$

E_{θ_0} is the expected value for a fixed θ_0. This expected value can be estimated by an average of M trials, $X_1, X_2, \ldots, X_M$:

$$\frac{c(\theta)}{c(\theta_0)} \approx \frac{1}{M}\sum_{i=1}^{M} \exp\{(\theta-\theta_0)^T s(X_i)\}, \tag{5.6}$$

θ_0 is still fixed. Similarly, the approximation of $\mathcal{L}(\theta) - \mathcal{L}(\theta_0)$ is

$$\mathcal{L}(\theta) - \mathcal{L}(\theta_0) = -\log \frac{1}{M}\sum_{i=1}^{M} \exp\{(\theta-\theta_0)^T s(X_i)\}, \tag{5.7}$$

and if we can sample random networks from the distribution P_{Θ_0} then we can calculate the approximation and thus the log-likelihood. (The $\mathcal{L}(\theta_0)$ term is constant for fixed θ_0, so it does not matter for us.)

The idea of the Markov Chain Monte Carlo algorithm is to create a Markov chain which has the same stationary distribution as the distribution we want to draw from. There are various methods to accomplish this, we don't discuss them here, although they are important for the practical applications of exponential random graphs.

The only remaining question on solving the exponential random graph model is the choice of θ_0. Theoretically it does not matter which θ_0 we choose, the estimated log-likelihood converges to the right value; this convergence can be however very slow if it is far from the correct solution of θ. One method is to calculate a maximum pseudolikelihood estimate for θ and use it as θ_0. The maximum pseudolikelihood method was originally proposed for the fitting of the exponential random graph model, but it turned out that it has deficiencies.

The exponential random graph model is a mathematically well grounded and very general tool for data-driven tasks. It does not, however, supersede our kernel-based approach in all cases, as it has some deficiencies.

1. While it is inherently dynamic, it is not currently clear how explicit network evolution data (i.e. *when* exactly vertices and edges were added to the network) could be taken into account for model fitting.

2. The Markov Chain Monte Carlo method might have slow convergence, especially if one cannot estimate the possible range of parameters when

doing exploratory network analysis. Nowadays, the method can work for graphs with up to a couple of thousand vertices. The kernel based approach usually works easily for millions of vertices.

3. While the method gives the “correlation” between the dynamic driving force (the structural/intrinsic properties) and the observed network(s) in the form of the θ coefficients, this is just a number and not a function as in the kernel-based case, thus important information is missing.

4. The framework assumes a form of the probability distribution over the ensemble of graphs and the effects are additive in the power of the exponential. No such form is assumed in the kernel-based model.

A form of exponential random graphs were introduced first by Holland and Leinhardt [1981]. See [Wasserman and Robins, 2005, Robins and Pattison, 2005, Snijders et al., 2006] for recent technical summaries. [Park and Newman, 2004] is an excellent introduction from the statistical physics perspective.

5.3 Generalized preferential attachment

Generalized preferential attachment is a simple method introduced by Roth [2005]. It is basically equivalent to our frequentist solution, without the frequentist iteration. It considers the network static along the measurement, thus no normalization takes place. According to our experiments, the absence of normalization tends to cause false results if the distribution of vertex types is not stationary.

5.4 Kronecker graphs

The stochastic Kronecker graph model, defined by Leskovec and Faloutsos [2007], just like every other random graph model, assigns probabilities to all possible graphs with a given size and it is defined via so-called Kronecker products of matrices.

The Kronecker product of an $(n \times m)$ matrix (U) and an $(n' \times m')$ matrix (V) is an $(nn' \times mm')$ matrix (S), defined as

$$S = U \otimes V := \begin{bmatrix} u_{1,1}V & u_{1,2}V & \cdots & u_{1,m}V \\ u_{2,1}V & u_{2,2}V & \cdots & u_{2,m}V \\ \vdots & \vdots & \ddots & \vdots \\ u_{n,1}V & u_{n,2}V & \cdots & u_{n,m}V \end{bmatrix}. \tag{5.8}$$

If we start with the parameter square matrix Θ containing elements from $[0, 1]$ and then create $\mathcal{P} = G_k$ as $G_k = G_1^{[k]} = G_{k-1} \otimes G_1$, $G_1 = \Theta$ then element $\mathcal{P}_{ij}$ gives the probability of having edge (i, j) in the Kronecker graph.

Leskovec and Faloutsos [2007] give a method for fitting the Θ parameter matrix to the network data by maximizing the probability that a parameter matrix of the given size reproduces the observed graph:

$$\arg\max_{\Theta} P(G|\Theta) \tag{5.9}$$

If $\mathcal{P} = \Theta^{[k]}$ and σ is a permutation of the vertices mapping them to the lines of the Kronecker graph adjacency matrix then the probability that σ and $\mathcal{P}$ generate G is given as

$$P(G|\mathcal{P}, \sigma) = \prod_{(u,v)\in G} \mathcal{P}[\sigma_u, \sigma_v] \prod_{(u,v)\notin G} (1 - \mathcal{P}[\sigma_u, \sigma_v]). \tag{5.10}$$

Then summing over all permutations is needed. They use a Metropolis algorithm [Gamerman, 1997] to draw from the permutation distribution. The Kronecker graph method can be used for static graphs, no time evolution is needed, rather, the fitted model is intended to be the average of all possible time evolutions.

They also show an efficient $O(|V|)$ method for calculating the log-likelihood of a graph, calculate the log-likelihood gradient and use a gradient descent method to find the best Θ parameter graph. The Bayes Information Criterion [Schwarz, 1978] is used for estimating the size, N_1 of the Θ parameter matrix.

The Kronecker graph method is effective and relatively simple, it has however some drawbacks, in our opinion. While it worked well on the examples shown by the author, it is unclear which properties of the network a fitted Θ matrix can reproduce well and which it cannot. It seems that the application of the method requires to "round" the number of vertices in the network to the closest power of the size of the model matrix Θ, as the process only generates graphs in which the number of vertices is the power of N_1.

6

Conclusions

6.1 Trait-based networks

The first part of the dissertation studied the effect of a specific kind of vertex correlation to the structure of networks. We defined a model of evolving social networks, where the connection probabilities depend on the traits of the vertices. To see the effect of the correlation introduced by the traits, we first examined a reduced, averaged model, in which each pair of vertices has the same δ connection probability (Sec. 2.2).

We calculated the degree distribution of the δ-based model by using a master-equation framework and showed that it is an exponential distribution. With a similar method we calculated the distribution of the maximal connected components, and found that usually it falls down exponentially, except near the phase transition threshold δ_{crit}, where it is a power-law distribution. Using generating functions we showed that the phase transition is always present in the model if $k \geq 2$.

Then, in Section 2.3 we numerically showed that although the full, trait-based model features the same qualitative behavior, the fine structure of the network is effected by the distribution of the traits and their connection probabilities. In particular, we showed that the degree distribution is a sum of exponential distributions, thus itself an exponential, and the tail of the distribution is dominated by the degree distribution of the vertex type with the highest attractiveness. Using the μ probability that two components will be merged by a new vertex, in the $k = 2$ case we showed that vertex correlations always decrease the phase transition threshold to a giant component.

6.2 Reverse engineering network evolution

The second, and main part of the dissertation aims to solve the inverse problem of network evolution: what are the parameters in the dynamical equations which fit to a given network best?

First, we defined a model framework, kernel-based networks (Sec. 3.1), where (in the simplest case) one single function gives the probabilities of the possible structural changes. This function depends on various properties of the vertices, these properties serve as the driving force of the evolution. We defined the goodness of a kernel function for a given network, allowing the comparison of alternative models for the same system.

Then we developed two methods for extracting the kernel function from network evolution data. The first, so-called frequentist method (Sec. 3.2) is based on a simple scoring model and involves the calculation of the leading eigenvector of matrix defined by the network evolution. We proved that the power iteration is always convergent for this matrix, if the network fulfills certain minimal requirements.

The second is a maximum likelihood method (Sec. 3.3) and finds the kernel function with maximum goodness. We showed that for most networks of interest it always has a unique solution, and the solution can be found with any optimization method capable of finding a local maximum of a nonlinear function. We showed that if the network has a stationary vertex type distribution, then the two methods yield the same result.

We applied the developed methodology to various networks (Sec. 4). In particular, using a time-dependent description, we showed that the preferential attachment exponent in the US patent citation network shows a sudden change around 1990, suggesting that the legal changes in the patent system had a serious effect on the dynamics of patent citations. We confirmed this finding by using other, more elaborated models. As another application, we showed that for every scale-free network, the best in-degree based kernel is linear preferential attachment, and similarly, for every network with exponential degree distribution, the best in-degree based kernel is uniform, non-preferential, i.e. random attachment.

Finally, we compared our methodology to others in the literature in Section 5.

A

Generating kernel-based networks

A.1 The partial prefix sum tree

TO CARRY OUT THE ERROR calculation of a model with respect to some network property, we need to be able to generate kernel-based networks of the same size as the original dataset under study. Although in some cases smaller networks are enough, depending on the particular model, usually it is easier to compare the (structural) properties of networks of the same size.

Generating kernel based networks involves generating random numbers from a general discrete probability distribution, usually one which is changing in time, as the types of the vertices change as well in time, and the kernel function giving the probabilities is a function of these.

Generating random numbers from a general discrete distribution is not very difficult, perhaps the simplest way to do it is the following. We assume that a single element of the set $a_i, \dots, a_N$ must be chosen and that the probability of choosing element a_i is given by p_i for all $1 \le i \le N$, $\sum_i p_i = 1$. The following simple algorithm can be used:

1. We preprocess the $p_1, \dots, p_N$ probabilities by calculating the cumulative sum of them: $q_0, q_1, \dots, q_N$, where $q_i = \sum_{j=1}^{i} p_j$.

2. We draw a uniform random number between zero and one: x.

$$\begin{tikzcd}
 & & & & & & & \sum_{i=1}^{8} p_i \arrow[dllll] \arrow[drrrr] \\
 & & & p_1+p_2+p_3+p_4 \arrow[dll] \arrow[drr] & & & & & & & & p_5+p_6+p_7+p_8 \arrow[dll] \arrow[drr] \\
 & p_1+p_2 \arrow[dl] \arrow[dr] & & & & p_3+p_4 \arrow[dl] \arrow[dr] & & & & p_5+p_6 \arrow[dl] \arrow[dr] & & & & p_7+p_8 \arrow[dl] \arrow[dr] \\
p_1 & & p_2 & & p_3 & & p_4 & & p_5 & & p_6 & & p_7 & & p_8
\end{tikzcd}$$

Figure A.1: Partial prefix sum tree for eight elements.

3. The q_i element for which $q_{i-1} \leq x < q_i$ is true is searched and the chosen element will be a_i.

Note that the p_i values do not necessarily need to sum up to one, i.e. there is no need to norm them, the only required trivial modification to the algorithm is that in step two a random number between zero and q_N must be drawn.

The first step can be done in $O(N)$ time, the second in $O(1)$ and the third in $O(\log N)$ as q_i are in increasing order and a binary search can be applied. Of course the preprocessed q_i values can be used for generating more a_i elements. As long as the p_i probabilities do not change, M random numbers can be generated in $O(M \log N + N)$ time. If the probabilities change often, like they usually do when generating kernel-based networks, then obviously a better algorithm or data structure is needed.

The partial prefix sum tree is a binary tree and for N elements to draw from it contains N leafs. The tree has $\lceil \log_2 N \rceil$ internal levels, and since it is an almost complete tree, it has $2^{\lceil \log_2 N \rceil} - 1$ internal vertices. (The internal levels of an almost complete tree form a complete tree and the leafs of the tree are left justified.) The leaf vertices contain the p_i values, whereas an internal vertex contains the sum of the values of its children. The root of the tree contains $\sum_{i=1}^{N} p_i$. See Fig. A.1

There are three operations defined on partial prefix sum trees:

1. Building the tree. This can be done in $O(N)$ time, as the total number of vertices in the tree is surely less than $3N$ and each element of the tree needs to be processed once only: the sum of the two children is written into every internal vertex, starting from the lower levels.

2. Choosing an element from the tree according to the actual p_i probabilities. This implies generating a uniform random number x, between zero and the value in the root vertex: $\sum_{i+1}^{N} p_i$. Then a search is performed in tree, starting from the root vertex, with initial target value $t = x$. If the current vertex is a leaf then we choose its corresponding element. If it is not a leaf and the value in its left child is bigger than t then the left child is chosen as the next search vertex. In the other case the right child is chosen and the t target value is updated by subtracting the value in the left child from it. As we only traverse the tree once, from the root down to a leaf vertex, this takes $O(\log N)$ time.

3. Updating the probability of an element, i.e. updating p_i. This requires updates in the ancestors of p_i's corresponding leaf vertex, and there are $O(\log N)$ such vertices, thus the operation can be carried out in $O(\log N)$ time.

Typically, if only $O(1)$ probabilities are updated in a time step while generating a kernel-based network then this data structure allows the generation to be done in $O(n \log N)$ time for a network in N vertices.

A.2 Citation networks

For citation networks it is quite straightforward to use the partial prefix sum tree. Each a_i element corresponds to a vertex in the network and we use $A(x_i)$, the kernel function values for the elements as p_i. We set $p_i = 0$ for the vertices which are not yet present in the current time step. Adding a vertex to the network requires only one update operation and it is done in $O(\log N)$ time. Changing the type of a vertex is also just an update operation and it is thus $O(\log N)$ time. If only $O(1)$ vertices are updated in each time step then this yields an $O(M + N \log N)$ algorithm for network generation. Let us see some concrete examples.

'Degree' model If the kernel function is based on the in-degree only then one update operation is needed for each edge, since an edge changes only the type of the cited vertex. Plus there is one update operation per vertex. The total time complexity is thus $O((M + N) \log N)$.

'Degree & age' model In this model the number of time steps passed since a vertex was added is also part of the properties vector, we call this property age. In addition to the changes caused by the increasing in-degree there are changes because of age. We assume that age is binned into some larger units and then for each vertex there are at most as many updates as the number of such units in the complete network. In other words 'age' grows linearly, so each vertex visits an 'age class' at most once. The time complexity is thus $O((M + cN) \log N)$, N is the number of vertices, M the number of edges and c the number of age bins.

'Recent degree' model Recent degree means the number of citations (incoming edges) acquired recently, not longer than a given w time steps ago. Unlike normal in-degree the recent in-degree is not necessarily increases with time, it may also decrease. It is true however that one edge can result at most two update operations: one when it increases the in-degree of a vertex and one when it decreases, when it goes out of date. The edges go out of date in the same order as they were added to the network, so keeping the edges in a queue helps deciding which edge goes out of date next in $O(1)$ time. The time complexity is thus the same as for the 'degree' model: $O((M + N) \log N)$.

'Recent degree & age' model The same applies here as for the 'degree & age' and the 'recent degree' models, the time complexity is $O((M + cN) \log N)$.

'Cited categories' model In this model the kernel function depends on some discrete categories of the potentially cited vertices. If these categories are time-independent then the time complexity is $O(M + N \log N)$, tree updates are needed only when new vertices are introduced to the network. If the vertex types are time dependent then the time complexity depends on how often they change, in the worst case all vertex types change in every time step resulting time complexity $O((M + N^2) \log N)$. Note that in this case using a partial prefix sum tree makes no sense at all, as with the cumulative sum based solution the worst case time complexity is slightly better: $O(M \log N + N^2)$.

A.3 Growing networks

For non-citation networks it is not enough to store a single element in the partial prefix sum tree, as N^2 choices are possible, any two vertices may be connected. Another approach is to store all the probabilities that any i-vertex connects to any j-vertex, this requires a tree with n^2 elements, n is the number of vertex types. In each time step however we might require $O(n)$ updates if the type of a vertex changes or a new vertex is added to the network. Adding a new edge requires just one updates, plus additional $O(n)$ if it changes the types of some vertices. All in all, in general we require $O((N + M)(n^2 \log n))$ steps.

B

Software tools: the `igraph` library

In this section, we briefly present the software tools implemented for the kernel based methodology. For the details see the `igraph` Reference Manual at `http://igraph.sf.net`.

The `igraph` library is an open source platform for graph algorithms, containing implementations for many convenience data structures and algorithms [Csárdi and Nepusz, 2006]. Lots of graph algorithms are already implemented in the library.

The library itself is implemented in C, some "external" algorithms not written by the `igraph` authors in C++, and it features high level interfaces to the Python [van Rossum, 1995], GNU R [R Development Core Team, 2007] and Ruby [Flanagan and Matsumoto, 2008] languages.

The kernel-based methodology is implemented in C and can be used currently from GNU R too, here we will introduce the latter. Both `igraph` and GNU R are portable to most systems, they can be easily installed on various MS Windows versions, all GNU/Linux flavors and Mac OSX versions. See the GNU R homepage at `http://www.r-project.org`. After GNU R is installed, the installation of `igraph` can be carried out by simply typing

```
> install.packages{igraph}
```

on most systems. The current version of GNU R is 2.6.1, the `igraph` version at the time of writing is 0.5. See also the `igraph` homepage for newer versions, documentation, support, mailing lists, etc.

Function name	*Description*
`growing.random.game`	Growing random graph generator, $A(x) = 1$ for all x.
`barabasi.game`	Nonlinear preferential attachment model, can generate based on recent citations too. $A(r) = r^\alpha + a$. (‘`d`’, ‘`r`’)
`aging.barabasi.game`	Nonlinear preferential attachment plus aging, $A(r, a) = (r^\alpha + c)a^\beta$. (‘`d`’, ‘`r`’, ‘`a`’)
`cited.type.game`	Generator based on the type of the cited vertex. All forms of kernels. (‘`e`’)
`citing.cited.type.game`	Generator based on the types of the citing and cited vertices. All forms of kernels. (‘`e`’, ‘`i`’)
`lastcit.game`	Generator based on the time (number of vertices) passed since the last citation. Can handle all forms of kernels. (‘`l`’)

Table B.1: Currently implemented kernel based generators in `igraph`. If the form of the kernel is restricted, that is given too. The identifiers at the end give the possibly relevant vertex properties of the generators.

There are two large groups of kernel-based functions in `igraph`: kernel-based generators and kernel-based measurement functions. In the future versions of `igraph` all generator functions will begin with the ‘`evolver.`’ prefix (the dot can be included in R identifiers and it is commonly used instead of the underscore to separate words). All measurement functions have the prefix ‘`revolver.`’ (Reverse EVOLVER).

See Table B.1 for the kernel-based generator functions.

The measurement functions have two big classes, one for the frequentist method, see Section 3.2 and one for the maximum likelihood method, see Section 3.3. Maximum likelihood methods have the prefix ‘`revolver.ml.`’.

Each of the implemented different vertex properties has a one letter identifier, and the *key* of the measurement function is the concatenation of the lower case identifiers in alphabetic order. E.g. the measurement based

Id.	*Description*
‘`a`’	Age of the vertices, the number of times passed since the vertex was added to the network, binned into larger units.
‘`d`’	The (in-)degree of the vertices.
‘`e`’	The type of the potentially cited vertices.
‘`f`’	Forest fire model, whether the vertex is a neighbor of the new vertex or not.
‘`i`’	The type of the citing vertex.
‘`l`’	Time passed since the last citation. Typically binned into larger units.
‘`p`	The number of papers an author has in an affiliation network.
‘`r`’	“Recent degree”. Number of citations gained recently, a time window size needs to be given.

Table B.2: Currently implemented vertex properties in `igraph`

on degree (identifier ‘`d`’) and age (identifier ‘`a`’) has the key ‘`ad`’, the frequentist measurement method based on degree and age is implemented in ‘`revolver.ad`’, the maximum likelihood based method can be performed by calling ‘`revolver.ml.ad`’. The various identifiers are listed in Table B.2.

Most functions work with citation networks, currently the exceptions are ‘`revolver.d.d`’ and ‘`revolver.p.p`’, these have general growing networks as their input. Notice that these function names include two keys, one for each of the participating vertices in the connection.

There are two more function classes within `revolver`, one for calculating the goodness of kernel functions, these functions have prefix ‘`revolver.error.`’ and one for fitting kernels with predefined shape (see Section 3.3.2), these include the keys in upper case and have a suffix defining the fitted form.

All currently implemented measurement functions are included in Table B.3.

Function names(s)	*Description*
`revolver.{ad, ade, adi, air, ar, d, de, di, e, el, il, ir, l, r}`	Frequentist measurement functions for citation networks.
`revolver.{d.d, p.p}`	Frequentist measurement functions for growing networks.
`revolver.error.{ad, ade, adi, air, ar, d, de, di, e, el, il, ir, l, r}`	Functions for calculating kernel function goodness.
`revolver.ml.{ad, ade, d, de, df, f, l}`	Maximum likelihood measurement functions for citation networks.
`revolver.ml.D.{alpha, alpha.a}`	Maximum likelihood fitting of predefined shapes based on in-degree: $A_1(d) = d^\alpha + 1$ and $A_2(d) = d^\alpha + a$.
`revolver.ml.AD.{alpha.a.beta, dpareto}`	Maximum likelihood fitting, predefined shaped, based on in-degree and age: $A_1(d, a) = (d^\alpha + c)a^\beta$ and double Pareto forms.
`revolver.ml.ADE.{alpha.a.beta, dpareto}`	Maximum likelihood method, predefined for, based on in-degree, age and the type of the cited vertex: $A_1(d, a, e) = c_e \cdot (d^\alpha + c)a^\beta$ and double Pareto forms.
`revolver.probs.{ad, ade, d, de}`	Calculating the probabilities of edge experiments based on a kernel function.

Table B.3: Currently implemented measurement functions in `igraph`.

B.1 An example session

We show here an example session using the R interface of `igraph`. We will generate a kernel based network using degree as the vertex property and do frequentist and maximum likelihood measurements on it.

First we load the `igraph` package.

```
> library(igraph)
```

Then generate a Barabási-Albert graph, but with nonlinear preferential attachment, the power is 0.8, the constant out-degree of the vertices is five. We simplify the graph, remove the possible multiple edges.

```
> ba <- barabasi.game(20000, power=0.8, m=5)
> ba <- simplify(ba)
```

Now comes the measurement, we use the frequentist method, based on in-degree, this is the "correct" model of the network, the goodness of the kernel is 0.62.

```
> freq.deg <- revolver.d(ba)
> (freq.deg$error[1] - freq.deg$error[2])/ecount(ba)
[1] 0.6217114
```

Let's use a maximum likelihood measurement now, the goodness is slightly higher now:

```
> ml.deg <- revolver.ml.d(ba, niter=10)
> (ml.deg$logprob - ml.deg$logmax) / ecount(ba)
[1] 0.6226385
```

Let's plot the measured kernel, the result is in Fig. B.1.

```
> plot(ml.deg$kernel, log="xy", xlab="in-degree",
>       ylab="attractiveness")
```

Finally, we fit the form $A(d) = d^\alpha + a$ to the network, to see whether the method is able to find the correct $\alpha = 0.8$ and $a = 1$ parameters.

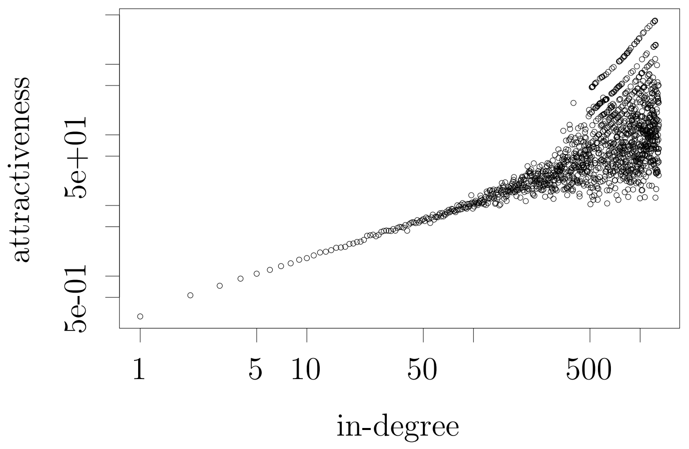


Figure B.1: Example degree based measurement using `igraph`

```
> sh.deg<- revolver.ml.D.alpha.a(ba, alpha=1, a=2)
> sh.deg["alpha"]
$alpha
[1] 0.801902
> sh.deg["a"]
$a
[1] 1.011969
```

Success. Please see the `igraph` documentation for more examples.

Bibliography

R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47, 2002.

R. Albert, H. Jeong, and A.-L. Barabási. The diameter of the world wide web. *Nature*, 401:130–131, 1999.

L. A. N. Amaral, A. Scala, M. Barhélémy, and H. E. Stanley. Classes of small-world networks. *Proc. Natl. Acad. Sci. USA*, 97(21):11149–11152, 10 2000.

B. Andrásfai. *Gráfelmélet (Graph Theory).* JATE Bolyai Intézet, Szeged, Hungary, 1997.

A.-L. Barabási. *Linked: How Everything is Connected to Everything Else.* Plume, 2004.

A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

A.-L. Barabási, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations. *Physica A*, 311: 590–614, 2002.

C. J. P. Belisle. Convergence theorems for a class of simulated annealing algorithms on rd. *Journal of Applied Probability*, 29:885–895, 1992.

G. Bianconi and A.-L. Barabási. Competition and multiscaling in evolving networks. *Europhysics Letters*, 54:436–442, 2001.

S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424:175–308, 2006.

M. Boguna and R. Pastor-Satorras. Class of correlated random networks with hidden variables. *Physical Review E*, 68:036112, 2003.

B. Bollobás. *Modern graph theory.* Springer, 2004.

B. Bollobás. *Random graphs.* Academic Press, 1985.

B. Bollobás and O. Riordan. The diameter of a scale-free random graph. *Combinatorica*, 24(1):5–34, 2004.

B. Bollobás and O. Riordan. Slow emergence of the giant component in the growing m-out graph. *Random Struct. Algorithms*, 27(1):1–24, 2005.

B. Bollobás, S. Janson, and O. Riordan. The phase transition in the uniformly grown random graph has infinite order. *Random Struct. Algorithms*, 26(1–2):1–36, 2005.

B. Bollobás, S. Janson, and O. Riordan. The phase transition in inhomogeneous random graphs. *Random Structures and Algorithms*, 31:3–122, 2007.

K. Börner, J. T. Maru, and R. L. Goldstone. The simultaneous evolution of author and paper networks. *Proc. Natl. Acad. Sci. USA*, 101:5266–5273, Apr 2004.

S. Breschi and F. Lissoni. Knowledge networks from patent data: Methodological issues and research targets. CESPRI Working Papers 150, CESPRI, Centre for Research on Innovation and Internationalisation Processes, Universita' Bocconi, Milano, Italy, Jan 2004. URL `http://ideas.repec.org/p/cri/cespri/wp150.html`.

M. Buchanan. *Nexus: Small Worlds and the Groundbreaking Theory of Networks.* Norton, W. W. & Company, Inc., 2003.

D. Callaway, J. Hopcroft, J. Kleinberg, M. Newman, and S. Strogatz. Are randomly grown graphs really random? *Phys. Rev. E*, 64:041902, 2001.

L. B. Ciric. A generalization of Banach's contraction principle. *Proceedings of the American Mathematical Society*, 45(2):267–273, 1974.

G. Clarkson. *Objective Identification of Patent Thickets: A Network Analytic Approach.* PhD thesis, Harvard Business School, 2003.

G. Csárdi and T. Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006. URL `http://igraph.sf.net`.

R. Diestel. *Graph Theory.* Springer, 2006.

S. Dorogovtsev and J. Mendes. Effect of the accelerating growth of communications networks on their structure. *Phys. Rev. E*, 63:025101, 2001.

S. Dorogovtsev, J. Mendes, and A. Samukhin. Growing network with heritable connectivity of nodes. cond-mat/0011077, 2000.

S. Dorogovtsev, J. Mendes, and A. Samukhin. Anomalous percolating properties of growing networks. *Phys. Rev. E*, 64:066110, 2001a.

S. Dorogovtsev, J. Mendes, and A. Samukhin. Giant strongly connected component of directed networks. *Phys. Rev. E*, 64:025101, 2001b.

S. Dorogovtsev, J. Mendes, and A. Samukhin. Generic scale of the "scale-free" growing networks. *Phys. Rev. E*, 63:062101, 2001c.

S. N. Dorogovtsev and J. F. F. Mendes. Evolution of networks with aging of sites. *Phys. Rev. E*, 62(2):1842–1845, 2000.

S. N. Dorogovtsev and J. F. F. Mendes. *Evolution of Networks. From Biological Nets to the Internet and WWW.* Oxford University Press, 2003.

P. Erdős and A. Rényi. On random graphs. *Publications Mathematicae*, 6: 290, 1959.

P. Erdős and A. Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci.*, 5:17, 1960.

P. Erdős and A. Rényi. On the strength and connectedness of a random graph. *Acta Mathematica Scientia Hungary*, 12:261–267, 1961.

G. Ergun and G. J. Rodgers. Growing random networks with fitness. *Physica A*, 303:261–272, 2002.

Federal Trade Commission. To promote innovation: The proper balance of competition and patent law and policy. Report, October 2003.

D. Flanagan and Y. Matsumoto. *The Ruby Programming Language.* O'Reilly, 2008.

D. Gamerman. *Markov chain Monte Carlo, stochastic simulation for Bayesian inference.* Chapman & Hall, 1997.

J. L. Gross. *Graph Theory and Its Applications.* Chapman & Hall, 2005.

B. H. Hall. Exploring the patent explosion. *Journal of Technology Transfer*, 30:35–48, 2005.

B. H. Hall, A. B. Jaffe, and M. Trajtenberg. The nber patent citation data file: Lessons, insights and methodological tools. In A. B. Jaffe and M. Trajtenberg, editors, *Patents, Citations, and Innovations: A Window on the Knowledge Economy.* MIT Press, 2003.

F. Harary. *Graph Theory.* Westview Press, 1994.

P. W. Holland and S. Leinhardt. An exponential family of probability distrinutions for directed graphs (with discussion). *Journal of the American Statistical Association*, 76:33–65, 1981.

D. Hunter. Exponential random graph models for network data, 2006. URL `http://www.stat.psu.edu/~dhunter/talks/ergm.pdf`.

A. B. Jaffe and J. Lerner. *Innovation and Its Discontents : How Our Broken Patent System is Endangering Innovation and Progress, and What to Do About It.* Princeton University Press, 2004.

H. Jeong, Z. Néda, and A.-L. Barabási. Measuring preferential attachment for evolving networks. *Europhys. Lett.*, 61:567–572, 2003.

K. Klemm and V. M. Eguíluz. Highly clustered scale-free networks. *Phys. Rev. E*, 65:036123, 2002.

P. Krapivsky, S. Redner, and F. Leyvraz. Connectivity of growing random networks. *Physical Review Letters*, 85:4629–4632, 2000.

J. Leskovec and C. Faloutsos. Scalable modeling of real graphs using kronecker multiplication. In *Proceedings of the 24th international conference on Machine learning*, pages 297–504, 2007.

J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187, 2005.

F. Liljeros, C. R. Edling, L. A. N. Amaral, H. E. Stanle, and Y. Åberg. The web of human sexual contacts. *Nature*, 411(907–908), 2001.

S. A. Merrill, R. C. Levin, and M. B. Myers, editors. *A Patent System for the 21st Century.* National Research Council of the National Academies, National Academies Press, 2004.

S. Milgram. The small world problem. *Psychology today*, 2(60), 1967.

M. Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Internet Mathematics*, 1:226–251, 2004.

M. Molloy and B. Reed. The size of the giant component of a random graph with a given degree sequence. *Combinat. Prob. Comput.*, 7:295–305, 1998.

M. E. J. Newman. Scientific collaboration networks.I. Network construction and fundamental results. *Physical Review E*, 64:016131, 2001a.

M. E. J. Newman. Clustering and preferential attachment in growing networks. *Physical Review E*, 64:025102, 2001b.

M. E. J. Newman. Assortative mixing in networks. *Phys. Rev. Lett.*, 89: 208701, 2002.

M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003a.

M. E. J. Newman. Mixing patterns in networks. *Phys. Rev. E*, 67:026126, 2003b.

M. E. J. Newman. Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, in press, 2005. URL `http://aps.arxiv.org/abs/cond-mat/0412004/`.

M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64:026118, 2001.

M. E. J. Newman, A.-L. Barabási, and D. J. Watts. *The Structure and Dynamics of Networks.* Princeton University Press, 2006.

J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer, 1999.

G. Palla, I. Farkas, I. Derényi, A.-L. Barabási, and T. Vicsek. Reverse engineering of linking preferences from network restructuring. *Physical Review E*, 70:046115, 2004.

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

D. J. d. S. Price. Networks of scientific papers. *Science*, 149:510–515, 1965.

D. J. d. S. Price. A general theory of bibliometric and other cumulative advantage processes. *J. Amer. Soc. Inform. Sci.*, 27(292–306), 1976.

R Development Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2007. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

S. Redner. Citation statistics from 110 years of physical review. *Physics Today*, 58:49, 2005.

G. L. Robins and P. E. Pattison. Interdependence and social processes: Generalized dependence structures. In P. Carrington, J. Scott, and S. Wasserman, editors, *Models and Methods in Social Network Analysis.* Cambridge University Press, 2005.

C. Roth. Measuring generalized preferential attachment in dynamic social networks. arxiv:nlin.AO/0507021, 2005.

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6:461–464, 1978.

T. A. B. Snijders, P. Pattison, G. L. Robins, and M. Handcock. New specifications for exponential random graph models. *Sociological Methodology*, 36(1):99–153, 2006.

B. Söderberg. A general formalism for inhomogeneous random graphs. *Physical Review E*, 66:066121, 2002.

B. Söderberg. Random graphs with hidden color. *Physical Review E*, 68 (015102), 2003a.

B. Söderberg. Properties of random graphs with hidden color. *Physical Review E*, 68:026107, 2003b.

B. Söderberg. Random graph models with hidden color. *Acta Physica Polonica B*, 34:5085–5102, 2003c.

R. Solomonoff and A. Rapoport. Connectivity if random nets. *Bulletin of Mathematical Biophysics*, 13:107–117, 1951.

B. Tadic. Dynamics of directed graphs: the world-wide web. *Physica A*, 293: 273–284, 2001.

S. Valverde, R. V. Solé, M. A. Bedau, and N. Packard. Topology and evolution of technology innovation networks. *Physical Review E*, 76:056118, 2007.

G. van Rossum. *Python Reference Manual*, 1995. CWI Report CS-R9525.

A. Vazquez. Knowing a network by walking on it: emergence of scaling. *Europhys. Lett.*, 54:430, 2001. cond-mat/0006132.

S. Wasserman and K. Faust. *Social network analysis methods and applications.* Cambridge University Press, New York, 1994.

S. Wasserman and G. L. Robins. An introduction to random graphs, dependence graphs, and p^*. In P. Carrington, J. Scott, and S. Wasserman, editors, *Models and Methods in Social Network Analysis*, pages 148–161. Cambridge University Press, 2005.

D. J. Watts. *Six Degrees: The Science of a Connected Age.* W. W. Norton & Company, 2003a.

D. J. Watts. *Small Worlds : The Dynamics of Networks between Order and Randomness.* Princeton University Press, 2003b.

D. J. Watts and S. H. Strogatz. Collective dynamics of small world networks. *Nature*, 393:440–442, 1998.

H. S. Wilf. *Generatingfunctionology.* Academic Press, London, 2nd edition, 1994.

H. Zhu, X. Wang, and J.-Y. Zhu. Effect of aging on network structure. *Phys. Rev. E*, 68:056121, 2003.