Preferential Attachment with Fitness Model

Rutger Mauritz, Huan Wu & Jop Zwienenberg

November 2020

1 Introduction & motivation

In this project, we will elaborate on the most important characteristics of a Preferential Attachment Model with Fitness (hereafter called PAF model), being the distribution of the vertex degrees in relation to its fitness values. We will do this by first mentioning the motivation for the PAF model and its relation to the classical Preferential Attachment (PA) model. In Section 2, we will then proceed with mentioning the most important theoretical results regarding the degree distribution of the PAF model, as provided by Borgs et al. [6]. In this section, we will also numerically verify and analyze these results by means of a Python implementation. We end this document with Section 3, where we show the connection between a continuous time branching process (CTBP) and the PA(F) model, which allows us to explain how a citation network should be modeled.

1.1 Preferential attachment model

The *Preferential Attachment* (PA) model proposed by Albert and Barabási [1] is a dynamic model that starts as a connected network of m_0 nodes. At each time step $t > 0$ a vertex v joins the network by adding a fixed number of $m \leq m_0$ links to the vertices that are already present in the network. This is done in a probabilistic way, such that the probability of connecting to an already existing vertex is proportional to its degree. Formally, the probability p_i that the new node links to node i at time t is given by

$$p_i = \frac{d_i}{\sum_j d_j}, \tag{1}$$

where d_i is the degree of node i before time t and where we divide by the sum of all degrees in the network before time t. This mechanism implies that vertices joining the network, would connect to high-degree vertices with larger probability than to low-degree vertices, accounting for hubs. By interpreting the degree of a vertex as e.g. the popularity of an individual, the PA model seems a good model to describe the well-known, 'rich get richer', phenomenon, which can be seen in both social networks, citation networks and the world wide web network [1].

1.2 Preferential Attachment model with Fitness

Despite its success in predicting certain graph structures, the PA model has one unsatisfactory assumption, being that the popularity of a vertex i is only resembled by the degree d_i of a vertex. This assumption leads to a phenomenon called, *'first-mover-advantage'*. This phenomenon states that vertices which enter the network earlier tend to have higher degrees than vertices which enter the network at a later time, meaning that the popularity of an individual is fully dependent on the time it entered the network. This highly conflicts with real-world networks.
A solution to the flawed assumption of PA models that incorporates this vertex quality is proposed by Bianconi and Barabási [2]. In their *Preferential Attachment with Fitness* (PAF) model, each vertex gets assigned a certain quality parameter, called *fitness*, drawn from a pre-defined distribution. This fitness parameter scales the degree of the vertex this parameter belongs to as time progresses, such that the popularity of an individual is now also determined by a pre-defined, external factor.
The PAF model is defined as follows (using the notation from [6]).

Definition 1.1 ***PAF model*** *Let $\mathcal{F} \subseteq \mathbb{R}_+$ be a set of fitnesses with $\mathcal{Q}$ a distribution over that set, the fitness distribution. The PAF model begins with one vertex with fitness $f \in \mathcal{F}$ drawn according to $\mathcal{Q}$ and a self-loop on that vertex. At every time step t, a new vertex gets added to the graph, whose fitness is again picked according to $\mathcal{Q}$. This vertex is attached to an old vertex v with probability proportional to $f_v \cdot d_{v,t-1}$, where f_v represents the fitness of vertex v and $d_{v,t-1}$ represents the degree of vertex v at time $t-1$. The graph at time n is denoted with $G_n = (V_n, E_n)$, where V_n is the set of vertices and E_n the set of edges.*

Below, one can see some realizations of such a PAF model, that we created by means of a Python implementation, see the appendix Sect. 4.1, in combination with *Cytoscape* [9].

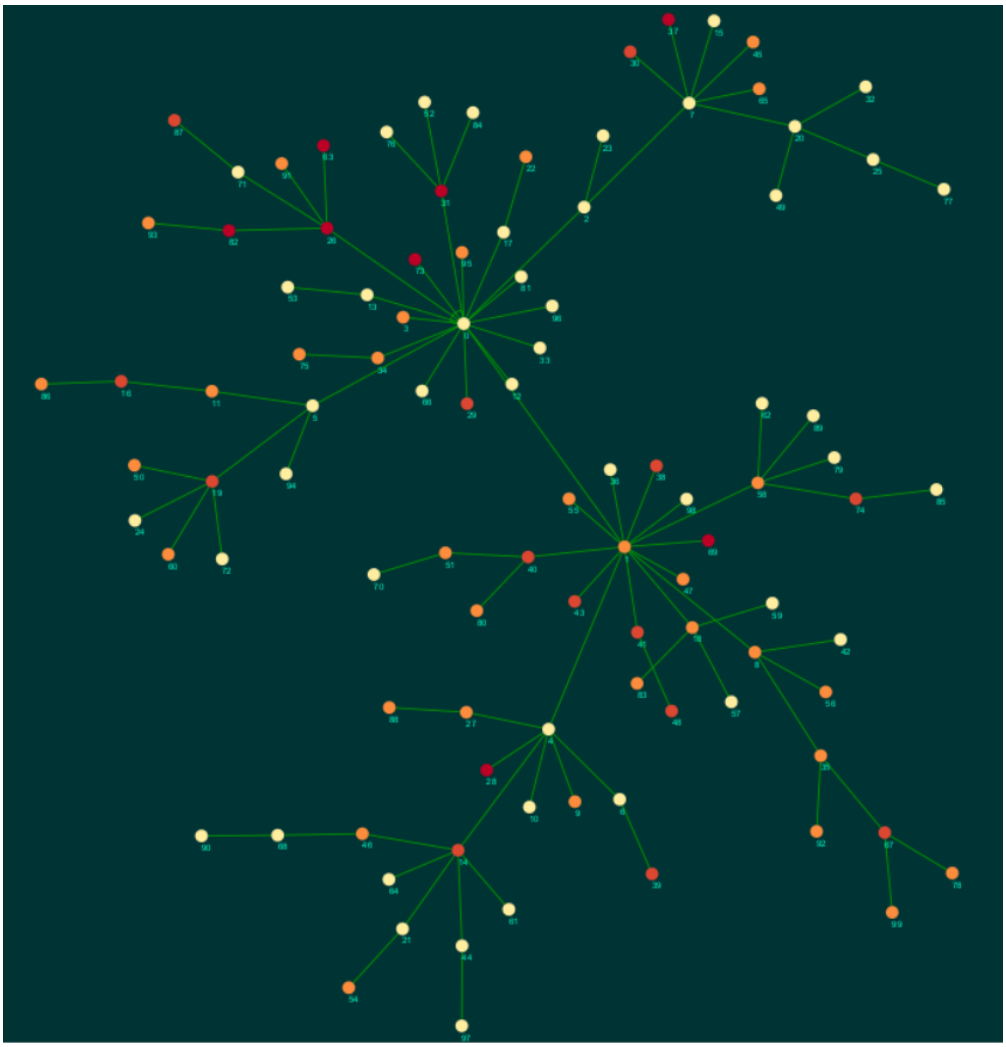


Figure 1: PAF graph with $n = 100$ and $Q = [0.4, 0.3, 0.2, 0.1]$

In Figure 1, a darker colour of a node means the node has a higher fitness value. In this relatively small example, we see the emergence of clusters or 'hubs' centered around nodes with a high fitness value, even though the evidence for this phenomenon is not very convincing yet. The latter is because of the small n and the relatively small value of $J = |Q|$. Note also the self-loop on vertex 0.

In Figure 2, however, the emergence of hubs is very much visible (although one can not deduce if the clusters are centered at nodes with a high fitness value, this will be proven later in this section).

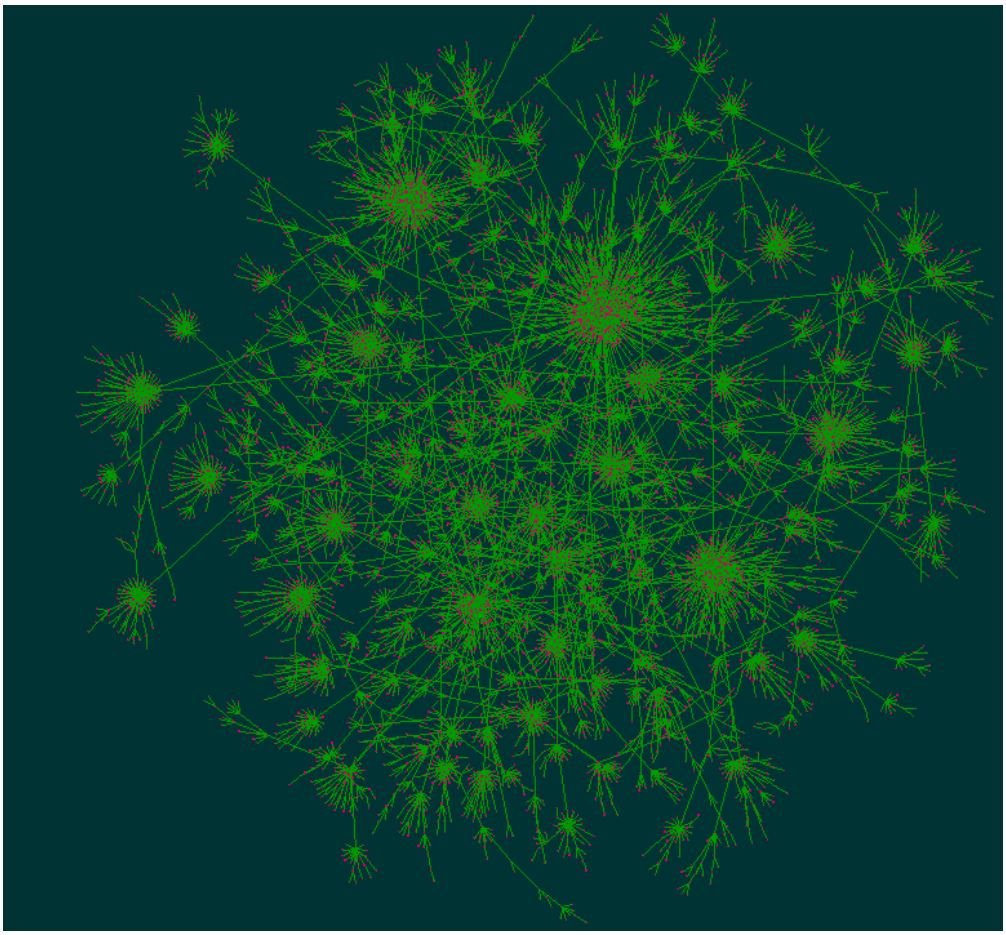


Figure 2: PAF graph with $n = 10000$ and
$Q = [0.3, 0.2, 0.1(\times 3), 0.05(\times 2), 0.025(\times 4)]$

1.3 Intuition in terms of social networks

We will now connect these models to real-life social networks. According to equation (1), preferential attachment implies that vertices joining the network would connect to high-degree vertices with larger probability than to low-degree vertices. Intuitively, the preferential attachment can be understood if we think in terms of social networks where people connect with each other. Here a link from A to B means that person A "knows" or "is acquainted with" person B. Heavily linked nodes represent famous/popular people with lots of relations. When a newcomer enters the community, he/she is more likely to become acquainted with one of those more visible people rather than with a relative unknown person. In other words, the probability of becoming acquainted with a particular person would be proportional to its degree.

Later, the PAF model works to address the issue of not taking the 'quality' of a person into account in the case of the PA model by introducing a fitness parameter. As previously mentioned, this parameter scales the degree of the vertex it belongs to as time progresses. In terms of social networks, again a link from A to B means that person A knows person B. It is reasonable to assume that when a person with a lot of quality in a certain area enters a certain community, the vertex representing him/her – as time progresses – has a much higher degree than most of the other person's vertices, even if that person entered the system much later. A good example is a young person with high footballing qualities that joins a big football club, coming from a small club. He entered the footballing community at a much later time than most professional footballers at that moment. However – as time progresses – he will be much more popular than most of the other average footballers.

Let us finally have a look at some other examples in other networks. Firstly, let the vertices represent the users of Spotify and the edges represent the subscription from one user to another user's profile. It is reasonable to assume that when a famous artist enters Spotify, the vertex representing his/her profile will - as time progresses - have much higher degree than most of the other user's vertices, even if that artist entered the system much later than those users. Secondly, the network of scientific citations; an interesting breakthrough paper will attract more citations than a marginal paper that appeared at the same time. Similarly, out of two bloggers, a celebrity will attract more attention. These examples show that the popularity of an individual is now also determined by a pre-defined, external factor, being independent of time.

2 Numerical verification of theoretical results for the degree distribution of PAF networks

In this section, we would like to numerically simulate, analyze and verify important theoretical results regarding the degree distribution of both the PA and the PAF model. We will do this based on theoretical results from the book *'Random Graphs and Complex Networks'* [7] and from the results from the paper of Borgs et al. [6]. When we talk about 'important theoretical results', we mean the following three areas to which these result apply:

1. The limiting degree distribution of the PA network.
2. The limiting behaviour of the fraction of edge endpoints per fitness value in PAF networks.
3. The limiting degree distribution within each fitness value in PAF networks.

For the first, we will mainly use [7], whereas for the latter two we will mainly use results from [6]. The analysis in this latter paper is applicable to discrete and continuous fitness distributions, as well as finite and infinite ones. The results from the paper are applicable to networks that are modeled by undirected graphs, but can be extended to directed graphs. **In this section, we will restrict ourselves to the case where the fitness distribution $\mathcal{Q}$ is discrete and finite and where the graph is undirected.** For this specific class, the most important results can be found in Section 3 of [6].

2.1 Results for PAF models with a discrete, finite $\mathcal{Q}$

We would like to mention the most important results from [6]. In this paper, Borgs et al. have derived several interesting results regarding the PAF network that we will elaborate on in this section.

The paper starts off with mentioning that, depending on the fitness distribution $\mathcal{Q}$, an evolving network can undergo one of the following behaviors, or *phases*:

- the *first-mover-advantage* phase, which results from flat fitness distributions and corresponds to the power-law behavior predicted by the classical PA model. An evolving network can undergo this behavior if the fitness distribution is discrete and finite.
- the *fit-get-richer* phase, in which vertices of higher fitness grow faster than those of smaller fitness; the behavior here is a power-law within each fitness value, but the tail exponent decreases as the fitness increases. An evolving network can undergo this behavior if the fitness distribution is discrete and finite.
- the *innovation-pays-off* phase, in which roughly speaking the competition for links results in a constant fraction of the links continuously shifting to ever larger fitness values; this fraction of links that "escapes to infinity" is independent of the network size and is determined by the fitness distribution. An evolving network can not undergo this behavior if the fitness distribution is discrete and finite. So we will not further consider this phase in the remaining part.

In this class, the set of fitnesses $\mathcal{F}$ can be described as $\mathcal{F} = \{f_j\}_{j \in J}$, where J is finite. This means that number of different fitness values is finite. Borgs et al. have derived three important results regarding PAF models in this class. They have done so by mapping the preferential fitness process to a finite generalized Pólya urn scheme. In Section 2.1.1, we mention their result regarding the degree distribution for PA models. Then, in Section 2.1.2 we mention their results regarding the distribution of edges across the fitness values. What follows is Section 2.1.3, where these aforementioned results are combined to obtain the degree distribution within each fitness value as derived by Borgs et al. Note that those three sections correspond to the Sections 3.1-3.3 of [6]. Finally, we note that the results from [6] regarding the PA model (Sect. 2.1.1) and regarding the PAF models (Sect. 2.1.2 and 2.1.3) are only valid for when each new vertex makes only 1 new connection ($m = 1$). In [7], however, a result for $m \geq 1$ is derived for the flat fitness distribution (Sect. 2.1.1) which we will elaborate on in our numerical simulation. We will not elaborate on the case $m \geq 1$ for the distributions

regarding the PAF models (Sect. 2.1.2 and 2.1.3). The actual numerical simulations/analysis corresponding to the aforementioned sections can be found in Sect. 2.2, 2.3 and 2.4, respectively.

2.1.1 Flat Fitness Distribution: First-Mover-Advantage

Suppose $J = 1$ (this is the classical PA model, since we only have 1 unique fitness value). Let $L_{n,k}$ denote the number of vertices of degree k at time n; set $\mu_1 = \frac{2}{3}$ and for $k \geq 2$ define

$$\mu_k = \frac{2}{3}\prod_{l=2}^{k}\frac{l-1}{l+2} = \frac{4}{k(k+1)(k+2)} \sim k^{-3}.$$

In particular, $\{\mu_k\}_{k\geq 1}$ is a power law with tail exponent 2. Then the first important result concerns the corresponding (limiting) degree distribution and is given in Theorem 2.1.

Theorem 2.1 *For all $k \geq 1$,*

$$\frac{L_{n,k}}{n} \xrightarrow{a.s} \mu_k \text{ as } n \to \infty.$$

2.1.2 Competition for Links across Fitness Values

Now suppose that $J = |\mathcal{F}| > 1$ and finite. Borgs et al. aimed to compute the limiting behavior of the random variables $M_{n,j}$, $1 \leq j \leq J$, corresponding to the number of edges with an endpoint of fitness f_j at time n, counting twice edges with two endpoints of fitness f_j , i.e. the total degree of vertices of fitness f_j.

Now let us define q_j as the probability density of fitness value f_j. Then let λ_0 be the largest solution to the equation

$$\sum_{j=1}^{J}\frac{f_j q_j}{\lambda_0 - f_j} = 1, \tag{2}$$

where by monotonicity, $\lambda_0 \in (\max_j f_j, +\infty)$. Also for $1 \leq j \leq J$, set

$$\nu_j = \lambda_0\frac{q_j}{\lambda_0 - f_j}. \tag{3}$$

Then, the second important result is the characterization of the distribution of links across fitness values in terms of $\nu_j's$. This is result is given in Theorem 2.2.

Theorem 2.2 *For all $1 \leq j \leq J$,*

$$\frac{M_{n,j}}{n} \xrightarrow{a.s} \nu_j \text{ as } n \to \infty.$$

2.1.3 Finite Distributions: Fit-Get-Richer

In this section, we describe the result that Borgs et al. derived regarding the degree distribution of the PAF model with finite, discrete fitness distribution. For all $1 \leq j \leq J$ and $k \geq 1$, let $N_{n,(j,k)}$ denote the number of vertices of fitness f_j and degree k at time n. Then define λ_0 and $\{\nu_j\}_{j=1}^{J}$ as in Section 2.1.2.
Moreover, for all $1 \leq j \leq J$ and $k \geq 1$, set $\eta_{(j,k)}$ as follows:

$$\eta_{(j,k)} = \nu_j \cdot \frac{1}{k}\prod_{l=2}^{k}\frac{l}{l + \lambda_0 \cdot f_j^{-1}}. \tag{4}$$

In particular,

$$\frac{\eta_{(j,k+1)}}{\eta_{(j,k)}} = \frac{k}{k+1}\frac{k+1}{k+1+\lambda_0 f_j^{-1}} = 1 - \frac{1+\lambda_0 f_j^{-1}}{k}(1+o(1)), \tag{5}$$

for k large. This means that for a fixed j, $\{\eta_{(j,k)}\}_{k\geq 1}$ has tail exponent $\lambda_0 f_j^{-1}$. The last important result is then given in Theorem 2.3.

Theorem 2.3 *For all $1 \leq j \leq J$ and $k \geq 1$,*

$$\frac{N_{n,(j,k)}}{n} \xrightarrow{a.s} \eta_{(j,k)} \text{ as } n \to \infty.$$

2.2 Degree distribution PA

In this section we would like to show numerical results for the limiting degree distribution of the PA model. Now in Theorem 2.1, we can see that the limiting degree distribution of the PA model approaches a power law sequence. However, this stated result only holds for the case where m (the number of connections a new node makes) equals one. The reason for this is that in the paper by Borgs et al. [6], a model assumption is made that a new vertex makes only one new connection. In the book by Hofstad [7], similar results have been obtained, however these results also enable m to be larger than one. It is for this reason that we will use the stronger derivations from Hofstad.

Before we can proceed, however, we have to mention that the PA model proposed by Albert and Barabási [1] is not very detailed. It gives the following definition:

> To incorporate the growing character of the network, starting with a small number (m_0) of vertices, at every time step we add a new vertex with $m(\leq m_0)$ edges that link the new vertex to m different vertices already present in the system. To incorporate preferential attachment, we assume that the probability Π that a new vertex will be connected to a vertex i depends on the connectivity k_i of that vertex, so that $\Pi(k_i) = k_i / \sum_j k_j$. After t time steps, the model leads to a random network with $t + m_0$ vertices and mt edges.

This model description is very informal. For example, the model does not explain how the first edge is connected and does not tell whether the degrees should be updated after each attachment of a single edge. The latter is important when we want to derive numerical results for a PA model with $m \geq 1$. Because of this lack of details, several models have been suggested. We will use the model proposed by Bollobás and Riordan [3]. Also we will use notation and extra details from the book of Hofstad [7]. This leads us to the following model description:

Definition 2.1 ***PA model*** *The model $PA_t^{(m)}$ starts ($t = 1$) with one vertex v_1 with m self-loops (each self-loop adds 2 to the degree). At time $t+1$, we construct $PA_{t+1}^{(m)}$ from $PA_t^{(m)}$ by adding a single vertex v_{t+1} with m edges attached to it. These edges are attached sequentially with intermediate updating of the degrees. The eth edge is connected to vertex v_i, for $i \in [t]$, with probability proportional to $D_i(e-1,t)$, where, for $e = 1, \ldots, m$, $D_i(e,t)$ is the degree of vertex i after the eth edge incident to vertex v_{t+1} is attached. What's more, self-loops are allowed such that the eth edge incident to v_{t+1} is attached to v_{t+1} itself with probability proportional to $D_{t+1}(e-1,t)+1$.*

Then the main result for this subsection follows from Theorem 8.3 in [7] (where we take $\delta = 0$) and will be rephrased below.

Theorem 2.4 *(Degree sequence in preferential attachment model). Fix $m \geq 1$. There exists a constant $C = C(m) > 0$ such that, as $t \to \infty$,*

$$P\Big(\max_k |\frac{L_{t,k}}{t} - p_k| \geq C\sqrt{\frac{\log t}{t}}\Big) = o(1),$$

where $L_{t,k}$ is as in Sect. 2.1.1 ($n = t$) and

$$p_k = \frac{2m(m+1)}{k(k+1)(k+2)}, \quad k \geq m.$$

One should note that for $m = 1$, the sequence p_k is exactly the same as the sequence μ_k in Sect. 2.1.1. From now on we will use $t = n$, just as in Theorem 2.1. Note that this is possible because being in time $t = n$, is equivalent to the size of the network being n, as at each time step $t \geq 1$, 1 vertex is added to the graph.

2.2.1 Simulation

In the appendix, Sect. 4.1 and Sect. 4.2, one can find the Python code that we used for simulating PA networks and the code that we used to visualize the corresponding degree sequences. For each realization of a PA network, we visualized the degree sequence $\{\frac{L_{n,k}}{n}\}_{k\geq m}$ by means of a normalized bar plot, where each bar k represents the frequency of degree k in the network. Below, we plotted such degree sequences corresponding to 4 realizations of PA models with $n = 50, 200, 1000$ and $10,000$ vertices respectively. For each realization, we fixed $m = 10$. What is more, for each plot, we added the power-law sequence p_k with $m = 10$ as defined in Theorem 2.4 and computed the distance between the bar plot and the p_k sequence by means of the Jensen-Shannon distance (JSD), a distance measure between two probability distributions, in this case $\{\frac{L_{n,k}}{n}\}_{k\geq m}$ and $\{p_k\}_{k\geq m}$ as in Theorem 2.4. This is the square root of the Jensen-Shannon divergence. Then the JSD between two probability vectors $\mathbf{p}$ and $\mathbf{q}$ is defined as:

$$JSD(\mathbf{p}, \mathbf{q}) = \sqrt{\frac{D(\mathbf{p}||\mathbf{m}) + D(\mathbf{q}||\mathbf{m})}{2}}, \tag{6}$$

where $\mathbf{m}$ is the pointwise mean of $\mathbf{p}$ and $\mathbf{q}$ and D is the Kullback-Leibler divergence.

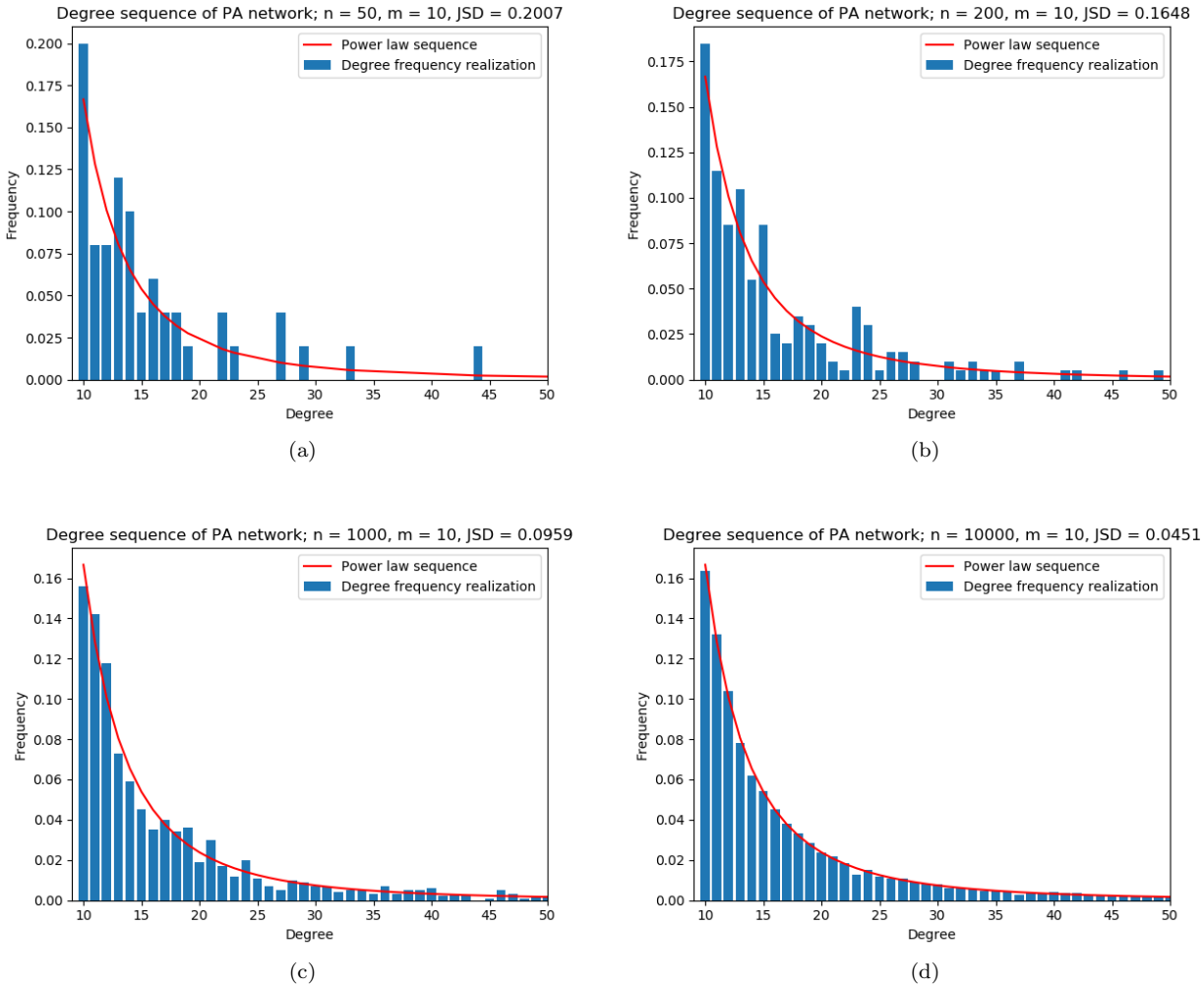


Figure 3: PA: degree distributions

As one can see, the empirical degree distribution of the PA networks get closer to the power law sequence

p_k as n increases. This is both justified graphically as well as by the fact that the JSD value decreases as n increases. Now as the PA model is a random model, the realizations differ from time to time. It is for this reason that - in order to be able to provide a mathematically justified conclusion - we repeat the above process an I number of times for each graph size $n = 50, 200, 1000$ and $10,000$, with m fixed and aggregate the results (determine the mean and the std. error). This gives us the following:

	n	JSD
$m = 1$	50	0.124 (0.0191)
	200	0.082 (0.0113)
	1000	0.050 (0.0060)
	10,000	0.024 (0.0020)

	n	JSD
$m = 4$	50	0.177 (0.0206)
	200	0.117 (0.012)
	1000	0.072 (0.0058)
	10,000	0.034 (0.0017)

	n	JSD
$m = 6$	50	0.197 (0.0237)
	200	0.132 (0.0129)
	1000	0.081 (0.0056)
	10,000	0.039 (0.0019)

Table 1: JSD (+std. error) between theoretical and empirical degree distribution, $I = 100$

One can clearly see that the conclusion that we drawed above, is valid in the general case. The JSD value decreases as n increases, which is true for any m. One should also note that as m increases, the convergence of the empirical distribution to the theoretical distribution is slower, which makes sense as the model complexity increases.

2.3 Competition for links across fitness values in the PAF Model

In this section, we will numerically verify Theorem 2.2, which is about the competition for links across fitness values. As is mentioned in Sect. 2.1.2, $M_{n,j}$ denotes the number of edges with an endpoint (vertex) having fitness value f_j at time n (or equivalently, in a network having n vertices). One should note that this is equivalent to the total degree of the vertices with fitness value f_j. In other words, $M_{n,j}$ gives us information on how much links are attracted by the class of vertices with fitness value j.

As mentioned in Sect. 2.1.2 we defined q_j as the probability density of fitness value f_j. We have assumed that that the fitness values in $\mathcal{F}$ are in ascending order, that is $f_j > f_k \quad \forall j > k$. Furthermore, in this simulation section we assumed that $f_j = j \quad \forall \quad 1 \leq j \leq J$.

Let us first start with a binomial distribution for the fitness values, with p being the success probability, $p \in [0, 1]$ and the number of trials being equal to J. Consequently, when the r.v. FV denotes the sampled fitness value, drawn from the set of fitnesses $\mathcal{F} = \{f_j\}_{j \in J}$, we have that the corresponding probability distribution is given by:

$$q_j = P(\text{FV} = j) = \binom{J}{j} p^j (1-p)^{J-j}, \quad 0 \leq j \leq J. \tag{7}$$

Note that the size of the support of this function equals $J + 1$ and the support includes $j = 0$. To prevent division by zero errors in our implementation, we added 1 to all of the fitness-values after the fitness values have been drawn. Summarizing, when writing $Q = Bin(J, p)$, we have that the fitness value of vertex v, $f_v \in \{1, 2, \ldots, J + 1\}$.

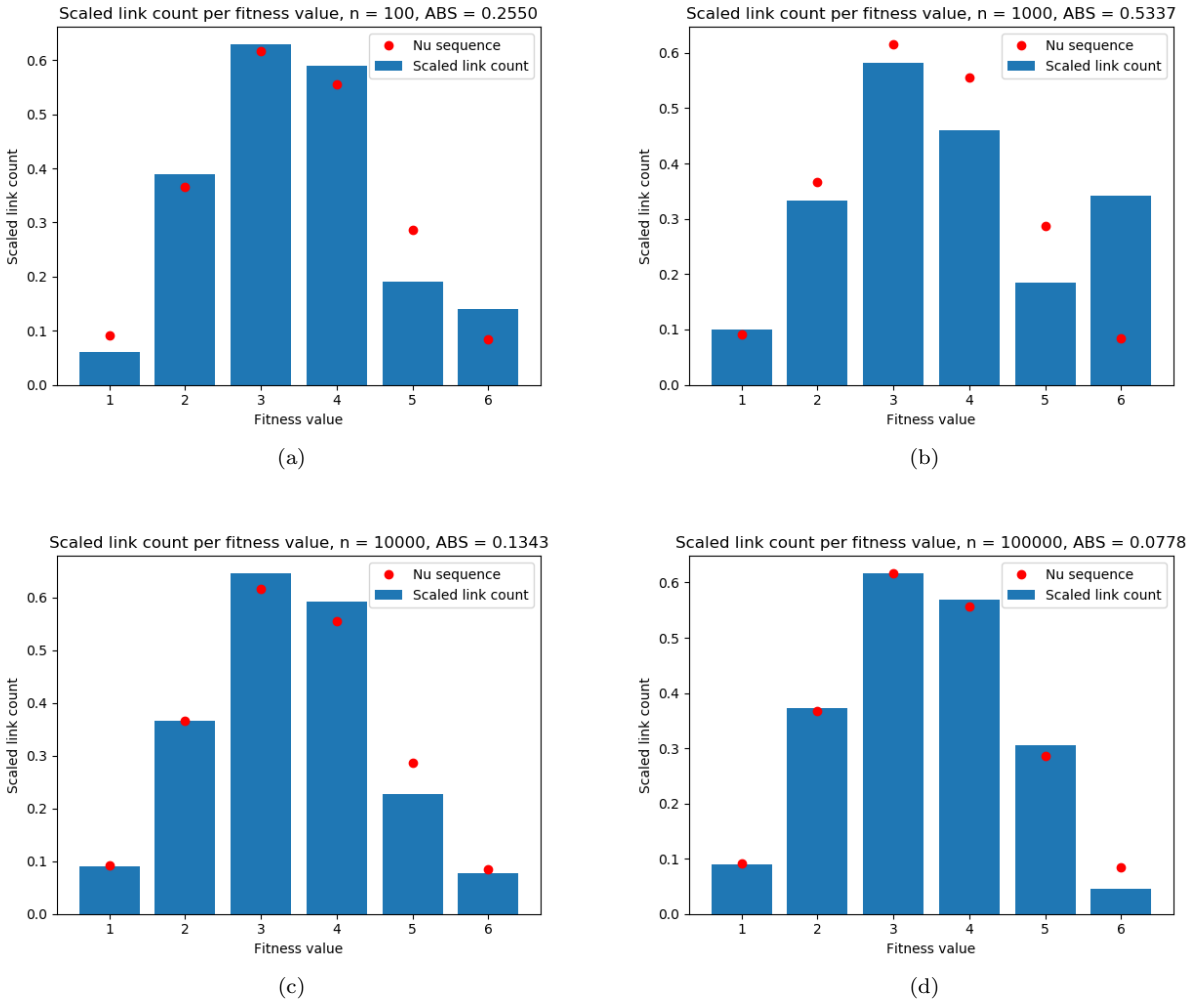
The distance metric that we will be using to measure the difference between the theoretical sequence $\{\nu_j\}_{j=1}^{J}$ and the empirical sequence $\{\frac{M_{n,j}}{n}\}_{j=1}^{J}$, is the sum of the absolute differences. That is, for two vectors $\mathbf{p}$ and $\mathbf{q}$ of length n, we define

$$ABS(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{n} |p_i - q_i|. \tag{8}$$

Finally, in order to solve Equation (2) for λ_0, we used the *solve* method from sympy

2.3.1 Simulation

The code that we used for this simulation can be found in the appendix, Sect. 4.1 and Sect. 4.3. If we simulate the link across fitnesses for Q being a binomial distribution with $J = 5$ and $p = 0.4$, we obtain the following results:



(a) (b) (c) (d)

Figure 4: PAF: links across fitnesses
$Q = Bin(5, 0.4)$

From these four plots, it is clearly visible that as n increases, the difference between the two sequences decreases, which is also shown via the decreasing ABS value. One should note that for these plots we used larger values of n than what we did for the degree distribution verification in Sect. 2.2.1. The reason for this is that the convergence is much slower. For the meaning of 'slower convergence', we refer the reader to the part below, where we investigate the influence of the cardinality of the support on the convergence.

Just as what we did in Sect. 2.2.1, we would like to repeat the above comparisons (with the same Q) for an I number of times and aggregate the results, as $M_{n,j}$ is a random variable. Because of computational complexity, we only do this for values of $n = 100, 1000, 10000$ and for $I = 50$. As one can see in Table 2, the above conclusion is valid in general.

n	ABS
100	0.619 (0.1917)
1000	0.308 (0.1175)
10000	0.143 (0.0601)

Table 2: ABS (+std. error) between theoretical and empirical sequence for link across fitnesses, $I = 50$

An interesting question is how the properties of the probability distribution Q influences these results. We decided to have a look at two properties in particular, that is the 'tailedness' of the distribution and the value of J.

Cardinality of the support:
For the latter, one should expect that as J increases, the convergence will be slower. Let j_1, j_2 be two fixed values of J s.t. $j_2 > j_1$. Then for some $\epsilon > 0$, define

$$n_1 = \underset{n}{\text{argmin}}\{n | ABS_{j_1}(\nu, \frac{M_n}{n}) \leq \epsilon\}$$
$$n_2 = \underset{n}{\text{argmin}}\{n | ABS_{j_2}(\nu, \frac{M_n}{n}) \leq \epsilon\},$$

where $ABS_j(\ldots)$ denotes the sum of absolute differences if $J = j$ is being used and where $\nu, \frac{M_n}{n}$ denote $\{\nu_j\}_{j=1}^J$ and $\{\frac{M_{n,j}}{n}\}_{j=1}^J$, respectively. We would then expect that $n_1 < n_2$.

In order to make a good comparison only based on the value of J, we use discrete, uniform distributions for Q with a varying J (note that in fact this is equivalent to having a normal PA model). When we write $Q = U_J$, we mean that Q is a discrete, uniform distribution with $P(FV = j) = q_j = \frac{1}{J}$ for all $1 \leq j \leq J$, where FV is as in Equation (7). Below in Table 3, one can see the results that we have obtained after having simulated for different values of J for $n = 100, 1000$ and 10000 and $I = 50$.

	n	ABS		n	ABS		n	ABS
$J = 5$	100	0.553 (0.2336)	$J = 6$	100	0.625 (0.2397)	$J = 7$	100	0.657 (0.2334)
	1000	0.211 (0.0990)		1000	0.251 (0.1012)		1000	0.284 (0.0621)
	10,000	0.079 (0.0378)		10,000	0.102 (0.0545)		10,000	0.128 (0.0569)

Table 3: ABS (std. error) between th. and emp. sequence for $Q = U_J$ and different values of J, $I = 50$

From Table 3, one can clearly see that our expectations turn out to be valid. As the value of J increases, we see that for a fixed value of n, the sum of absolute differences decreases.

Tailedness of the distribution:
In order to compare the results for distributions with different values for the tailedness, we will first need to find a way to quantify this tailedness. We do so by using the kurtosis measure. For r.v. X, we define the kurtosis

$$Kurt(X) = E\Big[\big(\frac{X-\mu}{\sigma}\big)^4\Big] - 3 = \frac{E[(X-\mu)^4]}{(E[(X-\mu)^2])^2} - 3 = \frac{\mu_4}{\sigma^4} - 3, \tag{9}$$

where μ_4 is the fourth central moment and σ is the standard deviation. We subtract 3 because this is the kurtosis value for the normal distribution. A kurtosis value higher than 0 means that is has fatter tails than a normal distribution and a kurtosis value less than 0 corresponds to having thinner tails. Now in order to draw a valid conclusion for the influence of the tailedness of Q, we will need to work with probability distributions that have the same cardinality of the support, but have a different shape. We decided to work with a Poisson distribution that is truncated from above. This allows us to have a finite, discrete probability

distribution for Q where we can change the tailedness just by varying the parameter λ. Say $f(x)$ denotes the pmf of the Poisson distribution and y equals the maximal value that the r.v. $X \sim Pois(\lambda)$ can achieve. The pmf of the truncated Poisson distribution is then given by

$$f(x|X \leq y) = \frac{g(x)}{F(y)}, \tag{10}$$

where $g(x) = f(x)$ for all $x \leq y$ and $g(x) = 0$ otherwise, and $F(x)$ is the Poisson cdf.

Now we have defined the measure for tailedness and the class of the probability distribution Q, we can show the effect of the tailedness on the convergence of $M_{n,j} \to \nu_j$. For $\lambda \in \{3, 4, \ldots, 14\}$, we simulated the links across fitness with fitness distribution Q_λ and fixed value of $n = 2000$ (because of computational complexity, we had to work with this relatively low value), where distribution Q_λ is a truncated Pois(λ) distribution with maximum value 5 (we also shifted the support by adding 1, just as we did for the binomial distribution). We aggregated the sum of absolute differences over the I simulations to obtain the mean ABS value together with the standard error. It turns out that for this sequence of values for λ, the higher the value, the higher the value of the kurtosis. It is for this reason that we plotted the kurtosis on the x-axis in Figure 5 below, where the first kurtosis value corresponds to $\lambda = 3$, etc. On the y-axis we plotted the mean of the ABS value over the I simulations together with an error band being $\pm$ std. error.

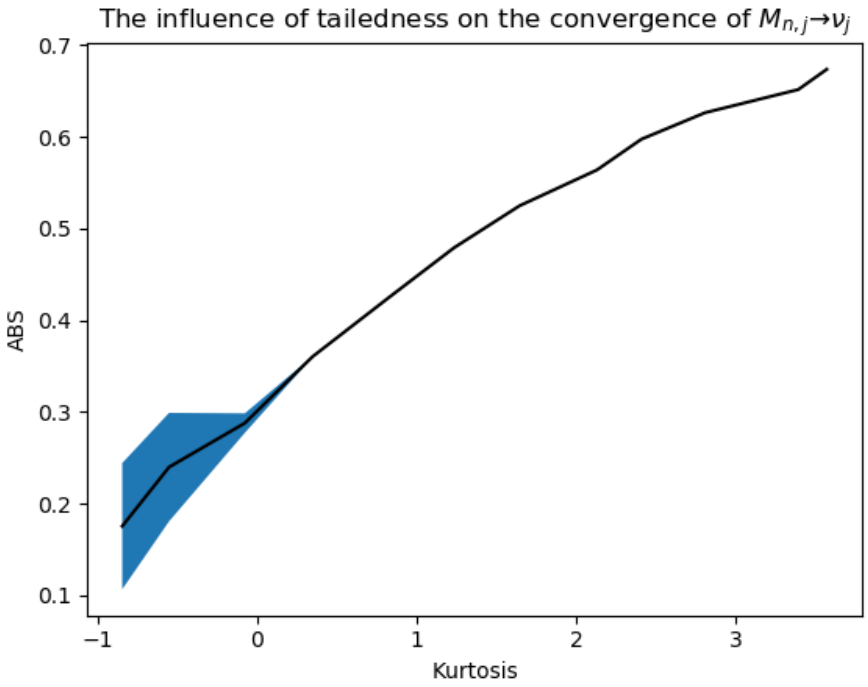


Figure 5: Influence of tailedness
$Q_\lambda = Pois(\lambda|j \leq 5)$ for $\lambda \in \{3, 4, \ldots, 14\}$

From Figure 5, one can clearly see that as the tailedness of the distribution increases, the sum of absolute differences increases, meaning that the convergence gets slowed down.

Summarizing, we can say that based on the above simulation results, we have shown empirically that the convergence of the links across fitnesses, as stated in Theorem 2.2, gets slowed down as $J = |\mathcal{F}| > 1$ increases and/or if the tailedness (as measured by the kurtosis) increases. One should note that this conclusion is drawn based on a small simulation with only one class of probability distributions for Q. A future research

step might be to verify this conclusion for more classes of probability distributions, which we could not do due to time limits.

2.4 Degree distribution within each fitness value in the PAF Model

The code that we used for this simulation can be found in the appendix, Sect. 4.1 and Sect. 4.4. In this section, we will numerically verify Theorem 2.3, which is about the degree distribution of the PAF model within each fitness-value (with finite, discrete fitness distribution). As is mentioned in Sect. 2.1.3, $N_{n,(j,k)}$ denotes the number of vertices of fitness f_j and degree k at time n. In other words, $N_{n,(j,k)}$ combines the results of Theorem 2.1 about the degree distribution and of Theorem 2.2 about the distribution of links across fitness values into one Theorem 2.3.

As a distance metric between $\{\frac{N_{n,(j,k)}}{n}\}_{k\geq 1}$ and $\{\eta_{(j,k)}\}_{k\geq 1}$, we used the ABS measure as defined in Equation (8) for each value of $1 \leq j \leq J$ and summed it up.

Important:
One problem that we encountered during the implementation of this numerical simulation is the fact that Equation (4) is said to hold for any degree $k \geq 1$. However, if $k = 1$, we end up with a product with an index l starting from 2 and ends at 1. Nowhere in the paper, we could find how this case should be interpreted. Because of limited time, we had to stick to the convention that this product then equals 1. Our advice is that for the numerical verification as is done in this section, it is good to figure out how Equation (4) should be interpreted if $k = 1$.

The results that we obtained from the numerical simulations were not very convincing to us. As an example, when we took $Q = Bin(5, 0.4)$ and took $n = 50,000$, we obtained the following result:

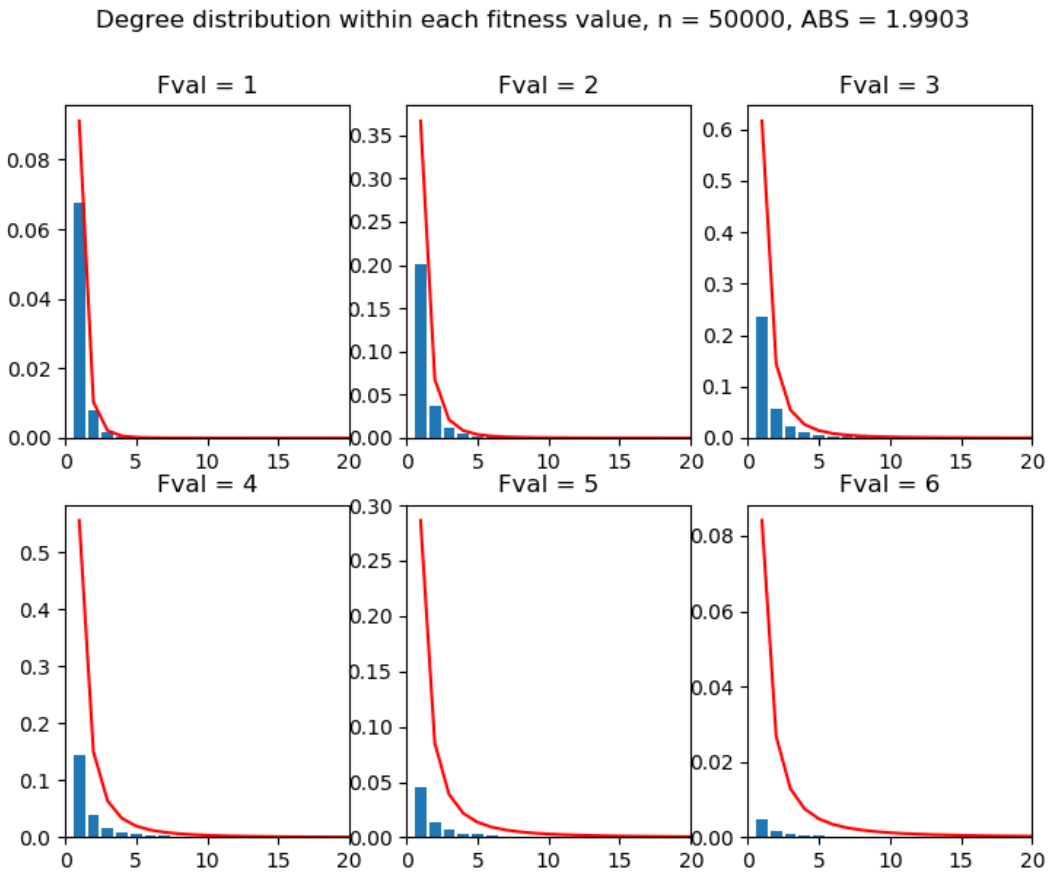


Figure 6: PAF: empirical degree distribution within each fitness value
$Q = Bin(5, 0.4)$

As one can see, the difference between the empirical sequence $\{\frac{N_{n,(j,k)}}{n}\}_{k\geq 1}$ and the theoretical sequence $\{\eta_{(j,k)}\}_{k\geq 1}$ increases as j increases. This behaviour has not become clear to us, as we would expect this to happen if the probability q_j of sampling a vertex j would decrease if j increases, which is not true however in the case $Q = Bin(5, 0.4)$, i.e. $Q = [0.0778, 0.2592, 0.34560.2304, 0.0768, 0.0102]$. Also, when we repeat the above simulation for $I = 50$ times for different values of n and aggregate the results, we do not see convergence as n increases, see Table 4.

n	ABS
1000	1.911 (0.0195)
5000	1.965 (0.0052)
10000	1.977 (0.0038)

Table 4: ABS (+std. error) between theoretical and empirical sequence
for the deg. distr. within each f_j

We observed similar results for different probability distributions Q. Because of limited time, we could not elaborate on this or try to find out what the reason for this behaviour might be. It is for this reason that we end this section with mentioning three possible reasons for the observed above:

1. The Python code that we used for the simulation is flawed.

2. We missed some details from the theory regarding Sect. 2.1.3 (e.g. the case when degree $k = 1$, as mentioned before).

3. Convergence only happens for even larger values of n that we used in our simulation. Unfortunately, because of the time complexity of the algorithms that we used, we could not simulate for higher values of n.

3 Modelling Citation Networks

In this section we will explain the connection between the preferential attachment (PA) model and continuous-time branching process (CTBP). Based on the results, we will state how citation networks can be modeled and why it is correct.

3.1 The Connection between PAs and CTBP

Continuous-time branching processes (CTBP) describe the evolution of a population whose individuals generate a random number of children according to certain birth processes $(V_t)_{t\geq 0}$ on $\mathbb{N}$. These processes are independent and identically distributed (i.i.d) across the individuals and defined by a sequence of weights $(w_k)_{k\in\mathbb{N}}$ describing the birth rates. In general, the time between the kth and $(k+1)$th jump (i.e. between the kth and $(k+1)$th birth) is an exponential random variable with parameter w_k for all individuals, so that the process is Markovian. The behaviour of the whole population is determined by this sequence.
The CTBP model can be used to understand the PA model in which the birth rates are affine functions (linear plus a constant), so we are going to investigate the connections between these two models. The connection is done by the Embedding Theorem, which states that the PA model has the same degree distribution as the CTBP model. In other words, we can model real-life networks, where preferential attachment occurs, by a CTBP model. An example of such a real-life network is a citation network. We will now look at the properties of citation networks.

3.2 How to Model Citation Networks?

3.2.1 Properties of citation networks

For a citation network model (graph model), vertices denote papers and the directed edges correspond to citations, i.e. if paper-i cites paper-j, then there is a directed edge coming from i to j. In order to model citation networks, we have to know the empirical properties of them so that we can come up with a model that fulfills these properties. We will first point out the properties analysed from three different types of citation networks, namely *Probability and Statistics (PS), Electrical Engineering (EE)* and *Biotechnology and Applied Microbiology (BT)* [8].

Real-world citation networks possess five main properties:

1. In Figure 7, we see that the number of scientific publications per year grows exponentially in time, since the y-axis is logarithmic. This could either be due to the fact that the number of journals that is listed in Web of Science grows over time, or that journals contain more and more papers.

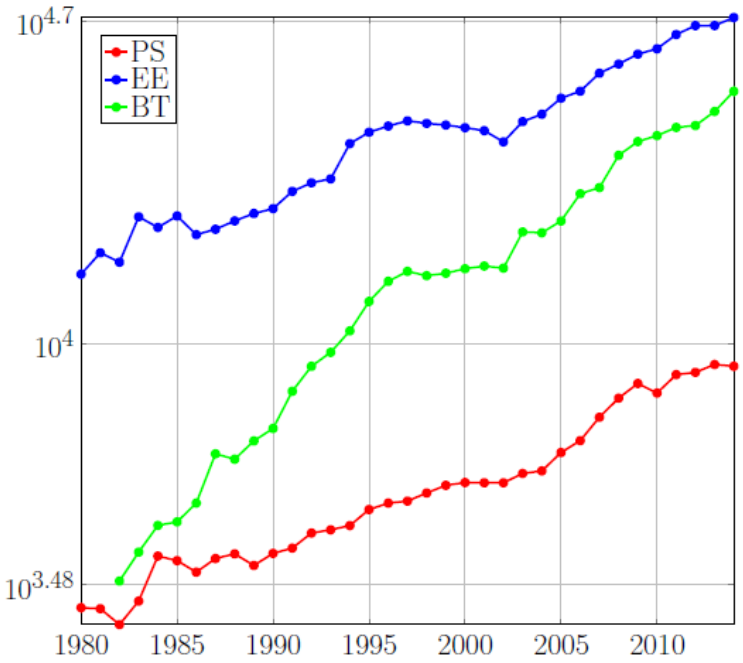


Figure 7: Number of publications per year with y-axis logarithmic.

2. Figure 8 shows that the log-log plot behaves 'linearly', so we can conclude that these data sets have empirical power-law citation distributions. Thus, most papers attract few citations, but few papers attract plenty of citations. The variability in the number of citations is rather substantial. The dynamics of the citation distribution is also of interest. This can be seen in Figure 9. We see a dynamic power-law over time, which means that at any time, the degree distribution is close to a power-law, but the exponent changes over time. In fact, the exponent decreases over time.

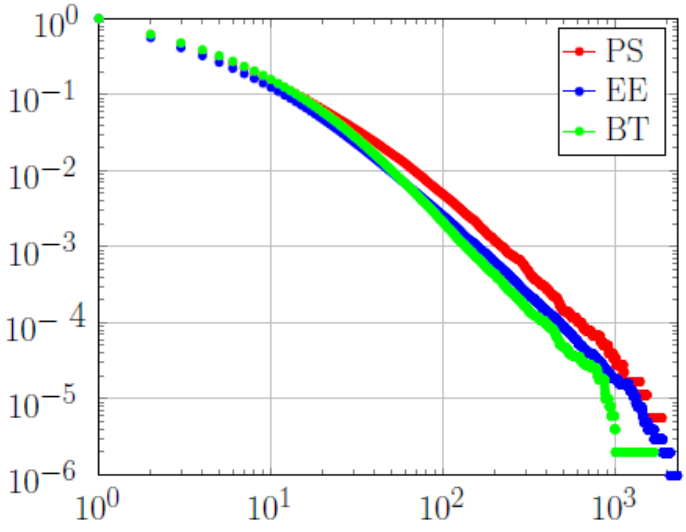


Figure 8: Log-log plot for the in-degree distribution in citation networks.

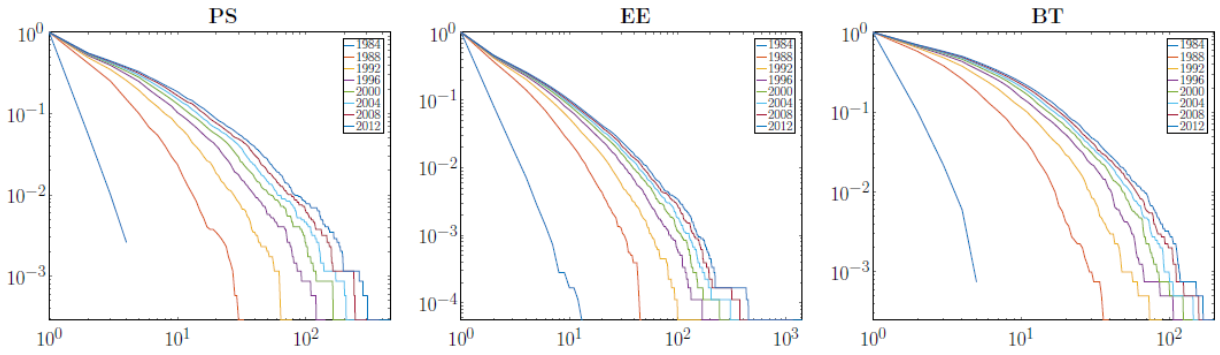


Figure 9: Log-log plot for the in-degree distribution for papers from 1984 over time.

3. In Figure 10, we see that the majority of papers stop receiving citations after some time, while few others keep being cited for longer time. This in-homogeneity in the evolution of node degrees is not present in classical PA models, where we have '*first-mover-advantage*' mechanism. Figure 10 also shows that the number of citations of papers published in the same year can be rather different, and the majority of papers actually stop receiving citations quite soon. In particular, after a first increase of receiving citations, the average increment of citations decreases over time, which can be seen from Figure 11. We observe that there is a difference in the aging effect, i.e. considering the *age of a vertex* in its likelihood to obtain citations, between PS data set and other two data sets. This is due to the fact that in PS, scientists tend to cite older papers than in EE or BT. Nevertheless, the average increment of citations received by papers in different years tends to decrease over time for all three data sets.

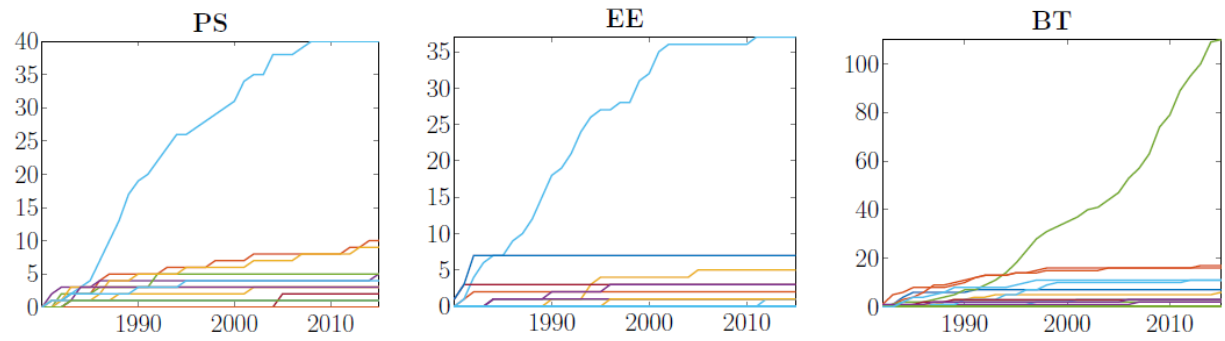


Figure 10: Time evolution for the number of citations of samples of 20 randomly chosen papers from 1980 for PS and EE, and from 1982 for BT.

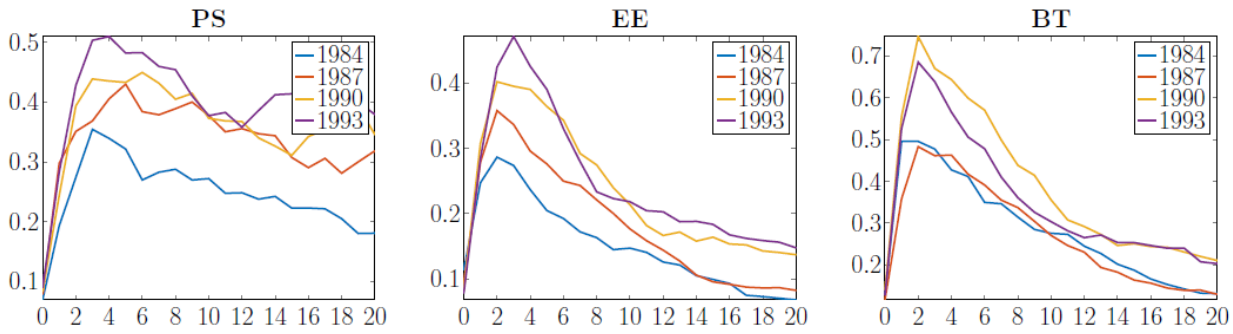


Figure 11: Average in-degree increment over a 20-year window for papers published in different years. PS presents an aging effect different from EE and BT, showing that papers in PS receive citations longer than papers in EE and BT.

4. Figure 12 shows the linear dependence between the past number of citations of a paper and the future ones. Each plot represents the average number of citations in the years 1992, 2005 and 2012 received by papers published in 1984 according to the initial number of citations in the same year. It can be concluded that at least for low values of the starting number of citations, the average number of citations received during a year grows linearly. This phenomenon suggests that the attractiveness of a paper depends on the past number of citations and should be defined as an affine function, i.e. a constant plus a linear function.

5. There is a log-normal distribution of the age of cited papers shown in Figure 13. In reference [8] the distribution of cited papers is plotted, looking at references made by papers in different years. A 20-years time window is used in order to compare different citing years. Notice that this log-normal distribution seems to be very similar within different years, and the shape is similar over different data sets.

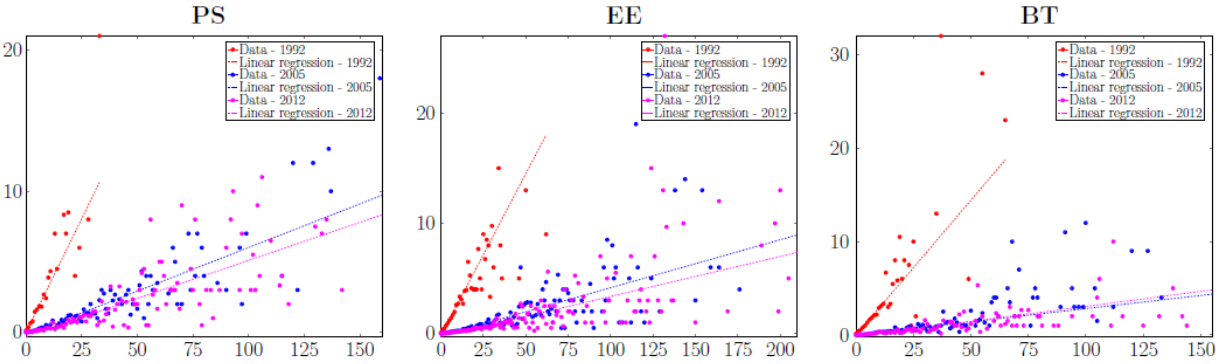


Figure 12: Linear dependence between past and future number of citations for papers from 1984.

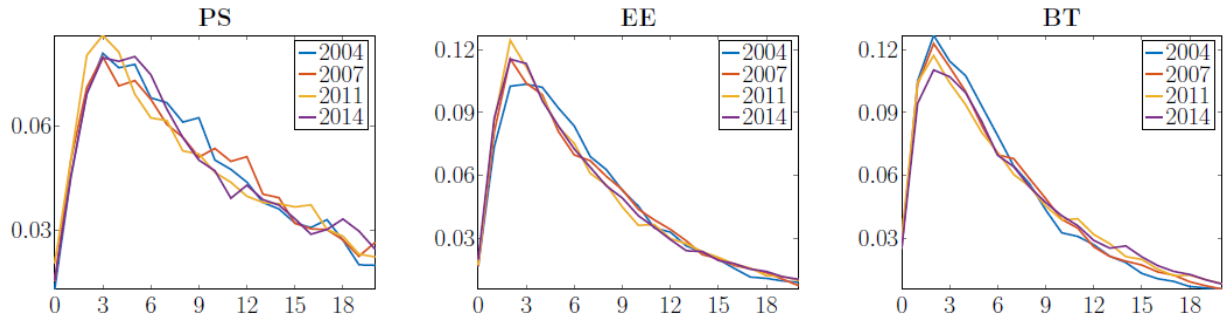


Figure 13: Distribution of the age of cited papers for different citing years.

3.2.2 Reason of using CTBP to Model Citation Networks

PA models can be used to model real-life networks which are dynamic and have power-law distribution, e.g. citation networks. By the Embedding Theorem [5], we know that the PA model has the same degree distribution as the CTBP model. Since the CTBP model is much easier to be analysed, we decide to use CTBP model fulfilling the properties mentioned before to model the citation network. We will now explain why we can use this Embedding Theorem (i.e., why we can use the CTBP model to model citation networks). Tree models, arising when new vertices are added with only one edge, lead to CTBP models. The degree distributions in tree models show similar qualitative behavior as for the non-tree setting (in particular PA models), while the analysis for tree model is much simpler. Motivated by this and the wish to understand the qualitative behavior of PA models with general aging and fitness, we use a CTBP model or tree setting. Indeed, the CTBP model at the nth birth time follows the same law as the PA model consisting of n vertices in which adding of new vertices is done as in Equation 1. The Embedding Theorem between CTBP models and PA trees is proved in reference [5].

3.2.3 How to translate empirical properties of citation networks into our CTBP model

We will explain how to translate the above empirical properties into our CTBP model. First, the CTBP model has the property to grow exponentially over time, as observed in citation networks. Secondly, the *aging* present in citation networks suggests that citation rates become smaller for large times, in such a way that typical papers stop receiving citations at some random point of time. The hardest thing is to translate the power-law degree sequence into our CTBP model. We notice that the attractiveness of citations of papers is influenced by many factors, such as the journals, the authors, the topics, etc., which cannot be quantified explicitly. Therefore, we introduce a random factor in the birth processes of the CTBP model, named *fitness*. To summarize, we want to define a CTBP model with both *aging* and random *fitness* that keeps having a power-law decay in the in-degree distribution.

3.3 Our CTBP Model

Let us consider a birth process $(V_t)_{t\geq 0}$ on $\mathbb{N}$, where every individual in the population is an i.i.d copy of the process $(V_t)_{t\geq 0}$ as explained earlier in this section. Then we consider the birth process defined by a sequence of weights $(w_k)_{k\in\mathbb{N}}$ describing the birth rates, i.e. the time between the kth and $(k+1)$th jump is exponentially distributed with parameter w_k.

The starting idea of our CTBP model of citation networks is that, given the history of the process up to time t,

$$\text{the rate of an individual of age } t \text{ and } k \text{ children to generate a new child} = Fw_k g(t),$$

where w_k is a non-decreasing weight function, F is a positive random variable called *fitness* and $g(t)$ is the integrable aging function.

Figure 12 implies w_k is affine, i.e. $w_k = ak + b$. Figure 13 implies the age of a cited paper is log-normal distributed. By normalizing such a distribution by the average increase in the number of citations of papers in the selected time window, we identify a universal function $g(t)$. This can be approximated by a log-normal shape of the form

$$g(t) \approx c_1 e^{-c_2(log(t+1)-c_3)^2},$$

for c_1,c_2 and c_3 field-dependent parameters. Therefore, the likelihood of a paper to receive new citations increases by having many citations and/or a high fitness, while it is reduced by age.

The fundamental theorem for the CTBP stated in Theorem 3.10 of paper [5] shows that, under some hypothesis on the birth process $(V_t)_{t\geq 0}$, the population grows exponentially in time, which fits Figure 7.

Let the ratio r be defined as

$$r := \frac{\text{number of individuals with } k \text{ children at time } t}{\text{size of total population at time } t}$$

Let T_α denote an exponentially distributed random variable with rate independent of the process $(V_t)_{t\geq 0}$. If there exists a unique positive α^* such that $\mathbb{E}(V_{T_{\alpha^*}}) = 1$ and $-\frac{d}{d\alpha}\mathbb{E}(V_{T_{\alpha^*}}) < \infty$, then by Theorem 3.10 and Definition 3.12 of paper [8], r converges almost surely to a deterministic value $p_k = \mathbb{E}(\mathbb{P}(V_{T_{\alpha^*}} = k))$. It is known from the literature [4] that in a CTBP model, p_k has a power-law distribution when the rates of jump depend only on a sequence of affine $(w_k)_{k\in\mathbb{N}}$. Since the proof of this is very involved, we will not perform this. Intuitively, it is also difficult to explain that power-laws can arise in CTBPs when the rates include *aging* and *fitness*. We cannot just say that because of the fact that most papers attract few citations and few papers attract plenty of citations we obtain a power-law, since citations of papers are also influenced by many external factors that affect the attractiveness of papers as mentioned earlier in this section. Nevertheless, if we apply Theorem 2.5 of reference [8] in some special cases, we can get a power-law distribution in our CTBP model. Corollary 2.6 of reference [8] ensures that our model has a dynamic power-law behavior if the fitness is exponentially distributed. Due to the limited time and material we can cover, we will not prove these theorems here.

References

[1] Albert-László Barabási and Réka Albert. "Emergence of scaling in random networks". In: *science* 286.5439 (1999), pp. 509–512.

[2] Ginestra Bianconi and Albert-László Barabási. "Bose-Einstein condensation in complex networks". In: *Physical review letters* 86.24 (2001), p. 5632.

[3] Béla Bollobás and Oliver Riordan. "The diameter of a scale-free random graph". In: *Combinatorica* 24.1 (2004), pp. 5–34.

[4] K.B. Athreya. "Preferential attachment random graphs with general weight function". In: *Internet Math* 4.4 (2007), pp. 401–418.

[5] Krishna Athreya, Arka Ghosh, and Sunder Sethuraman. "Growth of preferential attachment random graphs via continuous-time branching processes". In: *Proceedings Mathematical Sciences* 118 (Feb. 2007), pp. 473–494. DOI: 10.1007/s12044-008-0036-2.

[6] Christian Borgs et al. "First to market is not everything: an analysis of preferential attachment with fitness". In: *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. 2007, pp. 135–144.

[7] Remco Van Der Hofstad. *Random graphs and complex networks*. Vol. 1. Cambridge university press, 2016.

[8] Gerhard Woeginger Alessandro Garavaglia1 Remco van der Hofstad. "The Dynamics of Power Laws: Fitness and Aging in Preferential Attachment Trees". In: *J Stat Phys* 168.8 (2017), pp. 1137–1179.

[9] Institute for Systems Biology. *Cytoscape*. Version 3.8.2. Nov. 6, 2020. URL: https://www.cytoscape.org.

4 Appendix: Python code

In this section, we include all the Python code that is used for the numerical simulations of Sect. 2. This code can also be found on GitHub via https://github.com/RRMauritz/PAF-Network-analysis.

4.1 Network generators

Below one can find the methods that are used to generate the PA and PAF model.

```
def ba_graph_oi(n, m, seed=None):
    """
    OWN IMPLEMENTATION: use model from the CN book, page 256 (Bollobas & Riordan 2004)
    Here we use that delta = 0 (the Albert Barabasi model)
    :param n = the number of edges that we end the graph with (equivalent is the time when we end the process)
    :param m = the number of links each new vertex makes with the network
    Note 1: this algorithm allows for self-loops and multi-edges!
    Note 2: instead of using a probability vector, we use an occurrence list. This means that if we have two particles
    where particle 1 has twice as high probability to be drawn than particle 2 (p = [2/3, 1/3]), particle 1 will occur
    twice as often in this occurrence list than particle 2
    """

    # Start with 1 node, m self-loops, a self-loop adds 2 to its degree
    G = nx.MultiGraph()
    G.add_edges_from(zip([0] * m, [0] * m))
    # Add the first node (label = 0) 2*m times to the repeated-degree list:
    rep_nodes = [0] * m * 2
    # Set the source to 1 as we only have one vertex yet
    source = 1

    while source < n:
        # Loop over the m targets in each time step (intermediate scaling)
        for e in range(m):
            # this list adds the source each time new links is created -> p = D_(t+1)(e-1, t)+1 for connecting to itself
            rep_nodes_inbetw = rep_nodes[:]  # make hardcopy
            rep_nodes_inbetw.append(source)
            # Sample a target vertex from the rep_nodes_inbetw list that contains the extra source vertex
            target = list(_random_subset(rep_nodes_inbetw, 1, seed))[0]
            # Add an edge between the source and the target
            G.add_edges_from([(source, target)])
            # Add the source and vertex to the rep_nodes list as both their degree has increased by 1
            rep_nodes.extend([source, target])
        source += 1
    return G
```

Listing 1: Generating the PA network

```
def paf_graph(n, Q):
    """"
    Implementation of the Albert Barabasi graph with fitness
    :param n = the total vertices after the network simulation is over
    :param Q = a vector containing of the probabilities corresponding to each fitness value (ascending-> 1, 2, 3...)
    """

    # Start with a single vertex and self-loop on it
    G = nx.MultiGraph()
    G.add_edges_from(zip([0], [0]))
    # Sample n fitness parameters corresponding to the (future) vertices
    # Do this according to a probability distribution Q = [p1, p2, p3,...]
    fitnesses = choices(np.arange(1, len(Q) + 1), weights=Q, k=n)  # len(Q) +1 as right boundary not included
```

```
    # list of scaled degrees acting as prob. distr., 2*fitness value of first vertex (self
    loop -> deg x2)
    sc_deg = [2 * fitnesses[0]]
    # We start with the target being vertex 0
    target = 0
    # The new entering vertex, starting with label m0 (as Python is 0-based)
    source = 1
    while source < n:
        # Add edges from the source to the the m targets
        G.add_edge(source, target)
        # Increment the sc_deg list for the position of the target and source vertex
        sc_deg[target] += fitnesses[target]
        sc_deg.append(fitnesses[source])
        # Make a new list that transfer sc_deg to a probability distribution
        prob = np.array(sc_deg) / sum(sc_deg)
        # Sample m target nodes from the existing vertices with sc_deg as distribution
        target = np.random.choice(np.arange(len(sc_deg)), p=prob)
        source += 1
    return G, fitnesses
```

Listing 2: Generating the PAF network

```
def show_PAF(n, Q, graph_name, attr_name):
    file_loc_graph = os.path.join("Graphs", graph_name)
    G, fitness = paf_graph(n, Q)
    nx.write_graphml(G, file_loc_graph)
    file_loc_attr = os.path.join("Graphs", attr_name)

    a = np.stack(([i for i in range(n)], fitness), axis=1)
    np.savetxt(file_loc_attr, a, delimiter=",", fmt="%i", header='Node, Fitness')
```

Listing 3: Converting the PAF network to a .graphml file using Networkx

4.2 Simulating the degree sequence of the PA model

```
def deg_compare_PA(n, m, plot=False):
    """
    Compares the empirical degree distribution of the PA(n,m) model with the power law
    sequence
    Returns the JSD between the two and executes a plot depending on the parameter 'plot'
    :param n: the number of elements in the PA network
    :param m: the number of connections each new vertex makes
    :param plot: if True, this method will plot the degree and power law sequence
    """
    G = ba_graph_oi(n, m)  # Bollobas & Riordan 2004 Model!

    degrees = list(dict(G.degree()).values())  # list of degree values for all vertices
    deg_count = Counter(degrees)  # Count each occurrence
    deg_k = list(deg_count.keys())  # unique degrees (key)
    deg_v = list(deg_count.values())  # number of occurrences of each unique degree (value)

    deg_k, deg_v = zip(*sorted(zip(deg_k, deg_v)))  # sort the two together based on deg_k
    ascending
    deg_v = [v / n for v in deg_v]  # normalize the bars

    power = [(2 * m * (m + 1)) / (k * (k + 1) * (k + 2)) for k in
             deg_k]  # power law sequence (theoretical limiting distr.)

    jsd = jensenshannon(deg_v, power)  # Compute the Jenshen-Shannon divergence between the
    two

    if plot:
        plt.bar(deg_k, deg_v, label='Degree frequency realization')
        plt.plot(deg_k, power, 'r', label='Power law sequence')
        plt.xlim([m - 1, 50])
        plt.title('Degree sequence of PA network; n = %d, m = %d, JSD = %5.4f' % (n, m, jsd)
    )
```

```
        plt.xlabel('Degree')
        plt.ylabel('Frequency')
        plt.legend()
        plt.show()

    return jsd
```

Listing 4: Calculating and comparing the degree sequence of the PA model

```
def deg_compare_PA_sim(n, m, I):
    """
    This method generates a PA(n,m) network and computes the JSD distance between its
    empirical degree distribution
    and the known power-law sequence. It does this I times and returns the mean JSD value
    together with the standard error
    :param n: the number of elements in the PA network
    :param m: the number of connections each new vertex makes
    :param I: the number of simulations
    """
    jsd_values = np.zeros(I)
    for i in range(I):
        print(i)
        jsd_values[i] = deg_compare_PA(n, m, False)
    mean_jsd = np.mean(jsd_values)
    std_jsd = np.std(jsd_values)
    return mean_jsd, std_jsd
```

Listing 5: Simulating and aggregating the results of the degree sequence of the PA model

4.3 Simulating the link across fitnesses for the PAF model

```
def competition_compare_PAF(n, Q, lamb_0=None, plot=False):
    """"
    Simulates the link across fitnesses for the PAF model and applies the ABS measure to the
     empirical
    and theoretical sequence
    :param n = the number of vertices
    :param Q = the fitness probability distribution
    :param lamb_0: if None, calculate it, otherwise use the given value
    """
    # Create an instance of the PAF graph
    G, fitness = paf_graph(n, Q)

    # Compute lamb_0 based on Q => solve equation (if not given)
    if lamb_0 is None:
        Qs = Array(Q)
        l, j = symbols('l,j')
        eq = summation(j * Qs[j - 1] / (l - j), (j, 1, len(Q)))
        lamb_0 = max(np.abs(solve(eq - 1, l)))

    # For each fitness value, store the total degree
    degrees = list(dict(G.degree()).values())
    fit_link = {el: 0 for el in range(1, len(Q) + 1)}
    for i in range(len(degrees)):
        fit_link[fitness[i]] += degrees[i]

    # Make a list of the fitness values and the corresponding counts
    link_k = list(fit_link.keys())
    link_v = list(fit_link.values())
    link_k, link_v = zip(*sorted(zip(link_k, link_v)))  # sort ascending
    link_v = [l / n for l in link_v]  # scale by n

    # Compute the nu sequence
    nu = [lamb_0 * Q[j - 1] / (lamb_0 - j) for j in range(1, len(Q) + 1)]

    # Compute the sum of absolute differences
```

```
    abs_difference = np.abs(np.array(nu) - np.array(link_v))
    ABS = sum(abs_difference)

    # Make a bar plot if Plot=True
    if plot:
        plt.plot([i for i in range(1, len(Q) + 1)], nu, 'ro', label='Nu sequence')
        plt.bar(link_k, link_v, label='Scaled link count')
        plt.title('Scaled link count per fitness value, n = %i, ABS = %5.4f' % (n, ABS))
        plt.xlabel('Fitness value')
        plt.ylabel('Scaled link count')
        plt.legend()
        plt.show()
    return ABS
```

Listing 6: Calculating and comparing the links across fitnesses of the PAF model

```
def competition_compare_PAF_sim(n, Q, I):
    # Solve for lamb_0 and give it as argument to competition_compare_PAF
    # so that we do not have to recalculate it each time
    Qs = Array(Q)
    l, j = symbols('l,j', real=True)
    eq = summation(j * Qs[j - 1] / (l - j), (j, 1, len(Q)))  # The end term is included in
    the summation
    lamb_0 = max(solve(eq - 1, l))

    abs_values = np.zeros(I)
    for i in range(I):
        abs_values[i] = competition_compare_PAF(n, Q, lamb_0, False)
    mean_abs = np.mean(abs_values)
    std_abs = np.std(abs_values)
    return mean_abs, std_abs
```

Listing 7: Simulating and aggregating the results of the links across fitnesses for the PAF model

4.4 Simulating the degree distribution within each fitness value for the PAF model

```
def deg_compare_PAF(n, Q, lamb_0=None, plot=False):
    """
    Creates a PAF network and computes its degree sequence for each fitness-value
    :param n: the number of vertices in the PAF graph
    :param Q: the fitness probability distribution
    :param lamb_0: if None, compute its value, otherwise use the given value
    """
    # Create a PAF network
    G, fitness = paf_graph(n, Q)

    degrees = list(dict(G.degree()).values())
    # Group all vertices with the same fitness value and calculate the degree sequence
    withing each class
    fit_deg = {}
    for j in range(1, len(Q) + 1):
        degrees_j = [degrees[k] for k in range(len(degrees)) if fitness[k] == j]
        deg_count_j = Counter(degrees_j)
        deg_k_j = list(deg_count_j.keys())
        deg_v_j = list(deg_count_j.values())
        deg_k_j, deg_v_j = zip(*sorted(zip(deg_k_j, deg_v_j)))
        deg_v_j = [v / n for v in deg_v_j]
        fit_deg[j] = (deg_k_j, deg_v_j)

        # First calculate the nu_sequence
    if lamb_0 is None:
        Qs = Array(Q)
        l, j = symbols('l,j')
        # The end term is included in the summation, the first entry of Q corresponds to j =
     1
```

```
        eq = summation(j * Qs[j - 1] / (l - j), (j, 1, len(Q)))
        lamb_0 = max(np.abs(solve(eq - 1, l)))

    nu = [lamb_0 * Q[j - 1] / (lamb_0 - j) for j in range(1, len(Q) + 1)]

    # Then calculate the eta-sequence
    eta = {}
    for j in range(1, len(Q) + 1):  # because the fitness-value starts at 1
        eta[j] = [nu[j - 1] * (1 / k) * np.product(np.array([l / (l + lamb_0 * (1 / j)) for
    l in range(2, k + 1)]))
                  for k in fit_deg[j][0]]
    # Calculate the sum of absolute differences for each fitness class and add together
    ABS = sum([sum(np.abs(np.array(eta[j]) - np.array(fit_deg[j][1]))) for j in range(1, len
    (Q) + 1)])

    # Make a plot if Plot = True
    if plot:
        ncol = 3
        nrow = 2
        fig, axs = plt.subplots(nrow, ncol)
        fig.suptitle('Degree distribution within each fitness value, n = %i, ABS = %5.4f' %
    (n, ABS))
        axs = axs.ravel()
        for j in range(1, len(Q) + 1):
            axs[j - 1].bar(fit_deg[j][0], fit_deg[j][1], label='Empirical degree sequence')
            axs[j - 1].plot(fit_deg[j][0], eta[j], 'r', label='Theoretical degree sequence')
            axs[j - 1].set_title(
                'Fval = %i, ABS = %5.4f' % (j, sum(np.abs(np.array(eta[j]) - np.array(
    fit_deg[j][1])))))
            axs[j - 1].set_xlim([0, 20])
        plt.show()
    return ABS
```

Listing 8: Calculating and comparing the degree distribution of the PAF model withing each fitness value

```
def deg_compare_PAF_sim(n, Q, I):
    Qs = Array(Q)
    l, j = symbols('l,j')
    eq = summation(j * Qs[j - 1] / (l - j), (j, 1, len(Q)))
    lamb_0 = max(np.abs(solve(eq - 1, l)))

    abs_values = np.zeros(I)
    for i in range(I):
        print(i)
        abs_values[i] = deg_compare_PAF(n, Q, lamb_0, False)
    mean_abs = np.mean(abs_values)
    std_abs = np.std(abs_values)
    return mean_abs, std_abs
```

Listing 9: Simulating and aggregating the results of the deg. distr. withing each fitness value for the PAF graph