# Evaluation of a random graph model for complex networks

Bachelor Thesis
Informatik

Alexander Alt
Matriculation number 297694
alexander.alt@rwth-aachen.de

November 9, 2015

Supervisors:
Prof. Dr. Peter Rossmanith
PD Dr. Walter Unger

Advisor:
Dr. Somnath Sikdar

# Statutory Declaration

I hereby certify that all work presented in this bachelor thesis is my own, no other than the sources and aids referred to were used and that all parts which have been adopted either literally or in a general manner from other sources have been indicated accordingly.

Aachen, November 9, 2015

YOUR NAME

# Contents

# Chapter 1

# Introduction

There is an increasing interest in the behaviour of the graphs which represent social network. This can be merely a hobby, when one tries to to compute the "Six Degrees of Kevin Bacon" based on the IMDb, or more serious work, like the discussion if a network graph is sparse or not. What are social network graphs? That can be a graph which represents all connections of "friends" in Facebook, the database of all actors found in the IMDb, a scientist collaboration network, a citation network or an Email messages network.

For a moment, lets assume that the set of social network graphs are just another graph class like planar, chordal or $d$-degenerate graphs. Obviously we have the problem, that unlike the three latter graph classes, there is no known exact definition what constitutes a social network graph and what not.

Thus we have to discuss, what the differences between arbitrary graphs and social network graphs are. For example, assume that you have the task to create a model for the random generation of city maps. Clearly this task is rather simple, because the result will be a grid in the US, but should not much more complicated even for city in europe. This is because there is no major uncertainity, to distinguish between a a real road network (regardless of a single city or an entire country), and an arbitrary graph which does not represent a road network. A road network is typically connected, and more or less planar. Exemption are a few bridges or tunnels. Furthermore, vertices, which are representing crossroads or junctions, very rarely have degree $> 4$.

We need graph models, because a model allows us to test properties of the graph classes, which can not be confirmed or dismissed by just looking at some real graphs. Furthermore a model might help to answer open question about the underlying graphs: For example, the fact, if the graph class is sparse or not, heavily affects the running time of some algorithms on this kind of graphs, like the answer of the aforementioned question if a graph exhibits small world effect.

The goal of this thesis will be the Implementation and Analysis of the Forest Fire Model. We will

develop a generic graph generator, which creates graphs that models complex network graphs. We demand that the rules of the model should be intuitive from a sociologist point of view, this means that we do not want to use highly mathematical rules. Because, there is no sharp border for the graph class, it would be nonsense to "invent" a exact mathematical defintion for social network, such that a certain graph $G$ is a network graph, but graph $H$, which results from $G$ by insertion or deletion of a single edge, is not a social graph. This is in contrast to the three aformentioned graph classes (planar, chordal and $d$-degenerate graphs) which have exact boundaries. An example for a rule that does not make sense from a sociologist point of view: Every vertex is part of a clique that has size exactly $k$ and two cliques are connected by exactly $n$ links. On the other hand, we only want that the rules for the model come from sociologist, we do not demand, that the graph should be obtained from sociologist, because of the following reasons: When sociologists develop a friendship network within a school class, they tipycally make use of questionaires, which ask questions like "Who are your friends?". Unfortunately the results may be skewed due to boasters or shy people, who do not give honest answers. This may result in awkward occurrences, where person A likes person B, but person B loathes person A.

Furthermore, those networks from sociologists tend to be rather small, like one grade of school or around 100 people. However, small graphs are far too much dependable on randomness to get useful results for the derivation of a model. Therefore we enforce that our model and its analysis should be able to work on large graphs, for example up to one million vertices. For this reason we need a model that does not (seemingly) arbitrary adds and deletes vertices and edges from a graph, but steadily advances until a graph of desires size is created.

There are different types of complex network graphs that have to be distinguished: The first type are graphs, which connections are mutual, like Friendship connection on Facebook The second type are graphs, which connections are directed: Examples are Weblinks or a collaboration network from scientist. The third type are graphs, which connections are directed, but do not permit loops, like research citation networks. This last type most frequently occurs, when the data can be sorted on time, and old papers can not quote younger papers.

## 1.1 Overview

In chapter 2, we discuss the Erdős-Rényi Model and identify its shortcomings when it comes to the generation of network graphs. Furthermore, we conduct a survey on the characteristics and properties of a social network. In chapter 3, we finally present the Forest Fire Model. After we define the properties of the Forest Fire Model, we describe the IOmplemntation of the model, and discuss some results of the algorithm. In chapter 4, we try to answer open questions from from the implenetation in chapter 3, such as the fact there is an abrupt change in the behaviour of the model at a certain point. We show, that there exist a closed formula for the upper bound of the average degree and insertion depth of the Forest Fire Model.

# Chapter 2

# A survey of observed properties of real social networks

Assume one gets a graph where all labels are deliberately removed, how can we decide if this graph represents a social network or say, a road map In theory any possible graphs with $n$ nodes can have between 0 and $\binom{n}{2}$ edges. Therefore, there exist $2^{\binom{n}{2}}$ possible graphs, if we consider undirected graphs without no self-loops and multiple edges A graph with 5 vertices can therefore have up to $2^{10} = 1024$ possible permutations. A graph with 7 vertices already has up to $2^{21} = 2$ million possible permutations.

Therefore, it is impossible to compile an encyclopedia, which contains all possible graphs, even if we only want to list very small graphs. Even when analyzing a complete database of existing network graphs, for example Facebook graph, IMDb, collaboration networks and citation network, this is still a negligible quantity of all possible graphs and thus of all possible network graph. This means that our model must rely on some probabilities when creating the graph, and can not be fully deterministic.

## 2.1 Erdős-Rényi model

As an introduction, we will now discuss the Erdős-Rényi model (ER)[2]. Because the ER was created with no special graph class in mind, we do not expect that the ER is a model for networks graphs. However, we hope to identify some flaws in this model, which we can avoid in our own model.

The Erdős-Rényi model will create a graph with some properties, and we can check for each of these properties, if they make sense for a network graph. This allows us to build a refined model which takes into account our findings.

### 2.1.1 Definition of the Erdős-Rényi model

First we create a graph with $n$ vertices but no edges. For each pair of vertices add an edge between them with probability $0 \leq p \leq 1$, where $p$ is some constant which was defined by the user at the invocation of the algorithm. Recall, that both the average degree, as well as the degree distribution of an arbitrary vertex follows the binomial distribution. Therefore the resulting graph has (on average) $\binom{n}{2}p$ edges, and any vertex has on average $(n-1)p$ neighbors.

However it is easy to see, that any graph created with the ER has some properties [2]: Especially for small $p$ the resulting graph will not be connected at all, which collides with the observation that most network graphs have a giant component. If the graph created by the ER is not connected, then, on average, all components will have the same size. On the other hand if $p$ is chosen too large, then the graph will become very dense. If $p < \frac{1}{n}$ then (with high probability) all connected components have size $\mathcal{O}(\log n)$. If $p > \frac{1}{n}$ then (with high probability) there exists one giant component and all other connected components have size $\mathcal{O}(\log n)$.

It can be observed in network graphs, that the assumption that all edges have same probability is highly unlikely. As we predicted at the beginning of this section, this model is too simple, but we have some clues how to improve this model with regards to network graphs.

## 2.2 Properties of Network graphs

As we have already seen: A social network graph is rather complex and can by no means modeled by a simple model like the Erdős-Rényi model. Thus we interrupt our search for a model at this point and ask the question how the structure and characteristics of large network graphs look like. For this we identify several nontrivial measures, that where recommended by M. Newman [4] [5] which allow us to quantify a graph as objective as possible.

### 2.2.1 Transitivity

The most obvious and intuitive difference between a random graph and a social network is transitivity. This is analogous to the mathematical concept of transitivity, where for example the relation "$\leq$" is called "transitive". As an example "$a \leq b$" and "$b \leq c$" also imply that "$a \leq c$" holds. In the world of network, this means that when A knows B and B knows C, there is a high probability that A also knows C, because A and C were introduced to each other by $B$. See also the saying "A friend of my friend is also my friend". This principle will become important in the next chapter, when we consider how a person is introduced to new acquaintances. As a measure

we define the clustering coefficient $C$ of a graph as:

$$C = \frac{3 \times \text{number of triangles}}{\text{number of connected triples of vertices}} \tag{2.1}$$

where a "connected triple" means a triplet of vertices $(u, v, w)$, where edges $(u, v)$ and $(v, w)$ exist. They form a triangle $\Delta$ if the edge $(u, w)$ is also present. We have to multiply the numerator by 3 because for each triangle consisting of vertices $u$, $v$ and $w$ there exist the connected triples $(u, v, w)$, $(v, w, u)$ and $(w, u, v)$. In this way we ensure that $C$ can assume every value in the range between 0 and 1. $C$ represents the probability that two vertices that are neighbored to each others, share a common neighbor. What are typical values for network graphs. On the other hand for the Erdős-Rényi Model we expect, that $C \sim \frac{1}{p}$ holds, because this is the constant probability that the third edge of the triangle is also inserted by the Erdős-Rényi Model.

### 2.2.2 Degree distribution

In contrast to graphs created by the Erdős-Rényi model, for network graphs we expect a heavy tailed or right skewed distribution: That means, that there exist many vertices with small degree, while only a few vertices have high degree. However this does not necessarily mean, that the probability for degree 0 is most likely. We only demand that the degree distribution has a mode at some point, and shrinks asymptotically towards $P(X = k) = 0$ for $k \to \infty$. As a counterexample consider road maps. Junctions with 3 or 4 adjoining roads are common, but junctions with 5 or more roads are very rare. So there is an abrupt change between very common and highly unlikely degree sizes. This is obviously not the case for most network graphs

The degree distribution for network graphs heavily depends on the specific network. For example the network graph of all film actors, which are registered in the IMDb has a much higher average degree, than a a scientist coauthorship network, which can be explained as follows: For scientist there is a much stricter bound for the maximum number of people which can work together on a certain topic, because it seems strange if the there are 20 authors of a short paper which spans about 10 pages. For actors, on the other hand it is much easier to work with many different people, see for example the cast of Game of Thrones. Furthermore, it is common for scientist, that two people who succesfully presented a paper together, will also work together in the near future, while actors tend to change their supporting cast more frequently.

### 2.2.3 Power Law distribution

There exist an ongoing discussion among scientist if a power-law distribution is realistic for network graphs or not. There are two distinct (i.e. they are not equivalent) definitions of a power law distribution:

Variant A: (Densification Power Law)

$$|E| = |V|^{\alpha} \tag{2.2}$$

This results in an expected average degree of $|V|^{\alpha-1}$. This means that that the number of outgoing edges per vertex grows with increasing numnber of vertices. Because we expect every network graph to have between $|V|-1$ (number of edges in a tree) and $\binom{|V|}{2}$ (the number of edges in a complete graph) edges, we assume, that $\alpha$ lies in the range between $\frac{\log(|V|-1)}{\log(|V|)} \approx 1$ and $\frac{\log(|V|^2-|V|)}{\log(|V|)} \approx 2$. Of course for a single snapshot one can always find a apparent value with $\alpha = \frac{\log|E|}{\log|V|}$, but this value can be an absolute fluke. Therefore, in order to validate a power law distribution, we enforce that several snapshots of the same social network must exhibit the same value for $\alpha$, where we only allow a comparatively small margin of error.

The main problem is, that for a "proof" of a power law, one would have to observe a network under otherwise constant conditions, i.e. without changing environmental influences. Consider for example, the actors network. An actor from Hollywood in the 1930 did not appear in european movies, due to travelling constraints, which is clearly no longer an obstacle. Even when we have enough data available trying to determine if a notwork is Scale-Free is not easy, because of the following reason: We assume that a power law distribution holds. Then we expect the average degree to be $\frac{2|E|}{|V|} = \frac{2|V|^{\alpha}}{|V|} = 2|V|^{\alpha-1}$. Because $\alpha \in [1,2]$ holds, the average degree is almost a constant for small $\alpha$, and depends almost linear on $|V|$ for large $\alpha$, while for $\alpha \approx 1.5$ we expect that the average degree grows with $\sqrt{|V|}$. We are only interested in large network, because small networks (e.g. film actors in 1930) rely far to much on randomness. But still we work with very noisy data. Because $\log k$ only grows very slow for large $k$, it might be hard to distinguish between between power law and other distributions, especially when $\alpha$ is much closer to 1 than to its maximum 2. On the other hand point against: Consider the scientific collaboration network. Does the number of scientific collaborations depends solely on the number of scientist, our are there other constraints, which limit the upper bound of the number of collaborations? A major problem of this theory, if you consider a collaboration network, like, for example, of computer science coauthorship, the the number of persons you collaborate (write a scientific paper together) does not only depends on the number of scientist, but it is bounded by the amount of time you can spended on meeting new colleagues. This would mean that there would be a certain size of the network, where the number of ages for the bibliography becomes larger than the publication itself.

Variant B: (Scale-Free Networks)
A network is called scale free, if its degree distribution follows a power law. This means that when we represent the degree distribution on a log-log plot, the distribution seems to follow a straight line. A scale free network is sometimes called a prime example for a heavy tailed degree distribution. However a network does not need to follow a power law degree distribution, in order to exhibit a heavy-tailed degree distribution.

We define, that a network is scale-free, iff the following distribution can be observed:

$$p_k = C \cdot k^{-\alpha} \tag{2.3}$$

Taking the logarithmus naturalis of both sides, this equation can also be represented

$$\ln p_k = -\alpha \ln k + c \text{ where } C = e^c \text{ is some arbitrary positive constant} \tag{2.4}$$

In theory, this definition can be shown or dismissed with a single snapshot for any network graph. However, this may be difficult in practice, because the right-hand end of the data may become noisy, an effect that is even amplified on log-log plots. There are two possible solution, to solve this problem: We could use increasing bin width at the tail of the distribution, or we could use the cumulative distribution function, which is less susceptible to noisy tails.

Another problem is that the presumed power law might only be obeyed in a part of the degree distribution. Either because there is a deviation for small $k$ or because the is a cutoff that limits the tail of the distribution. Furthermore, in network where in-degree and out-degree differ (i.e a non symmetric network), we might observe a power law in one case but not in the other. Consider for example the citation network, where the out-degree of a vertex $x$ represents the size of the references of the corresponding paper. It would seem very awkward if a scientific thesis does not cite any other paper at all. Therefore $p_0 = 0$( read as: the probability that a paper cites no other papers is zero), merely indicates that the power law only starts at a certain $k > 0$. On the other hand, if a comparatively small article in some journal cites over 100 other papers, one could ask the question if the author even read through all those papers, while it would be completely normal for a dissertation thesis to refer to so many references. The in-degree of a citation network represents the number of times a certain paper gets cited by other papers. It is possible for a large paper, that it does not gets cited, not because it was outright bad, but only because it was just published last month. On the other hand, a very famous paper like the "Introduction to Algorithms", published by Cormen, Leiserson, Rivest and Stein in 1990, is cited over 9000 times according by citeser.ist.psu.edu alone. As a conclusion, we expect a power law distribution of the in-degree to be more common, than in the out-degree of the network. If we just apply common sense: The world World Wide Web graph (i.e. the links between webpages) might exhibit a power law distribution of Variant B. On the other hand only the in-degree of citation (quoted by) networks follows a power law, while the out-degree (quoting another paper) will most likely not follow a power law. The reason, for this is that citation is not a simple task, because you first have to analyze the paper, before you can start to cite. Thus there is a natural upper bound to the number of citations in a paper.

### 2.2.4 Degree correlations/centrality

While the previous measure try to quantify the graph as whole, the following measures try to find an order of "importance" for every single vertex. Our most simple intuition, might be, that when a

vertex has high degree, therefore it must be very important. Another interesting question arises, if there are two vertices with high degree, are they more likely to have a direct connection or not. In other words: Does the neighbor of a high-degree vertex also has a comparatively high degree. The problem with this degree correlation is the question, if having many connections really determines importancy. For example, your paper is not important, if you cite many unknown or obscure (i.e. rarely citated) papers. On the other hanf, it may be better sometimes to citate only a few, but meaningful, papers. For our example the IMDb of actors this means accepting about every trashy script, versus appearing only selected award winning movies. Especially if this measure is well known among participants of the network, this might easily lead to run up scores, like accepting every invitation on Facebook to have (apparently) more friends than some other person.

### 2.2.5 Eigenvector centrality

The Eigenvector centrality, is an improvement of the degree centrality: We want a centrality measure which rewards vertices, by marking them as important, that are neighbored by other important vertices. The basic idea here is, to define a vertex as important iff it has many important neighbors. Clearly this is a recursive definition of a centrality measure. Let $x$ is the vector, which stores the importancy of each vertex, $x = Ax$ where x is the adjacency matrix of the graph. Fortunately we can iteratively apply $x' = Ax$, albeit this computation should be solved with the help of a computer. A well known example for this measure is the Google PageRank algorithm.

### 2.2.6 Closeness centrality

The closeness centrality regards vertices as "important", if they have a small average distance to every other vertex. Thus we reward the most central points of the network. We use the following definiton for Closeness Centrality of a certain vertex $i$:

$$l_i = \frac{1}{n-1} \sum_{i \neq j} d_{ij} \tag{2.5}$$

where $d_{ij}$ is the geodesic distance of any shortest graph between vertices $i$ and $j$. However, we have to deal with the fact that the graph might contain more than one connected component. Here the distance becomes $\infty$, if we try to find a path from one component to the other. Therefore, it is recommended to use the inverse function:

$$l_i^{-1} = \frac{n-1}{1} \sum_{i \neq j} d_{ij}^{-1} \tag{2.6}$$

Hereby we eliminate the problem that $d_{ij} = \infty \Rightarrow l = \infty$, because the inverse of $\infty$ is 0, which contributes nothing to the sum. This measure is also reasonably well known in general public, through comparisons of the Erdős for scientists or the Bacon number for actors.

### 2.2.7 Network Resilience

Network resilience is a measure how the network behaves, when a vertex or a certain fraction of vertices is removed from the network, what happens with the average vertex-vertex distance. For example, the internet graph, heavily relies on resilience, because in otherwise, the network would break down, really fast. There exist two different tests [5], to examine the resilience of a network. The first test removes vertices at random, which causes the distance to increases very slightly. On the other hand, the second test removes vertices with highest degree first, which causes the distance to increase sharply. For example, when 1% of node are removed, the distance might double, because it is more likely that we remove central hubs of the network.

### 2.2.8 Community structure

Most social graphs exhibit many small clusters, which are possibly overlapping. Thus it makes sense to use these natural hierarchies in our development of social networks. As an example: The RWTH Aachen is divided into several faculties, with the "Fachschaft MPI" being on of them. The faculties are further subdived into individual departments and chairs or institutes. Obviously for two people from for the same chair, common sense tells as that the probability for them to publish a paper together, is much higher than fpor an arbitrary computer scientist and an physician. Furthermore, a young undergraduate student studying/working at a single institute is not supposed to collaborate on a scientific paper with a professor from a completely different Department As a note, "not supposed to" merely means that the probability that two people collaborate, declines with further separation, thus the fact that two people from different universities work together is unlikely (very small percentage) but not impossible. On the other hand for leading members of a institute it is far more common to collaborate with people from different institutes. As a conclusion, the more background information we have, the clearer can we define the hierarchy of the social network graph, that we want to create.

### 2.2.9 Clauset Hierarchical Structure

The aforementioned community structure has two disadvantages. The first disadvantage is, that only for a small labeled graph with well defined sociological boundaries one could define the boundaries of the different hierarchies by hand. On the other hand, for large graphs, this becomes increasingly unhandy. Therefore we need to find the hierarchies for large graphs automatically. Second disadvantage: We need to know the complete backgrounf of our data, in order to create the graph manually. However there might more other criterias, which also promote community structures but which are more subtle (mostly because we do not have enough data). For example, if we only have a list, who works at each department, but not precise information, like who is the head of department, or who is only just out of graduation. Other reasons for collaboration might be cultural backgrounds, common study time.

Clauset et al.[1] developed another approach, by not trying to derive the community structure using existing knowledge, but try to find the structures automatically. For this they use a Dendrogram, which is a tree structure that describes the community structure of the graph. Every vertex in the graph is represented by a leaf node of the dendrogram. The probability that two vertices are connected depends on their relatedness as defined by the dendrogram. This means that the probability of two people to collaborate oin athesis together is higher, if the path connecting them in the dendrogram is short. For large graphs they use Markov Chain Monte Carlo method to find the dendrogram which describes the graph best. At any step of the MCMC they rotate an random node of the dendrogram, and check if the resulting dendrogram explains the data better than the old one. The major drawback of this algorithm is, that it assumes that communities at each hierarchy level are disjoint and must not overlap, which is not necessarily the case in complex networks.

# Chapter 3

# The Forest Fire Model

## 3.1 Properties

As we have seen in the previous chapter, there are some requirements any model for social networks has to fulfill. It should ensure the properties of social networks, we developed in the previous chapter.

Obviously, it is evident that the model should create some heavy-tailed or right skewed degree distribution. Furthermore our research about centrality in the last chapter has shown, that due to the typical formation of clusters in a network graph (small world effect), the graph should have a small diameter, at least in comparison to random graphs created with the Erdős-Rényi model. Leskovec et al.[3] even suggest a shrinking diameter, but this will be very hard to implement, because it suggests not only a small world effect, which we already reasoned to be unlikely in chapter 2, but even an almost complete graph.

A we have seen, sometimes but not always a power law distribution can be observed in a network graph, thus to should be possible to get $|E| = |V|^{\alpha}$ in certain cases, but it should also be possible to create graphs with our model that do not exhibit a power law densification.

As we already demanded in the Introduction, the algorithm should be easy to explain and use: We do not want 100 different variables, but as few variables as possible. The algorithm should be quite fast for two reasons. As we will see, if the number of nodes is smaller than 1000, there is to much randomness or noise in the data, to analyze anything. On the other hand, real social networks also tend to be really big, and this is the kind of graphs we want to analyze with our model.

### 3.1.1 Informal Explanation of the Forest Fire Model

An author wants to create a bibliography for his/her paper. A first paper to cite is choosen. At this point we assume that the author has no or little knowledge of the topic, which means that he/she does not know which papers of the bibliography database deal with his topic. Thus he only chooses an arbitrary paper, and not necessarily the most famous or most helpful paper. Why does the author not choose the paper with most links? The reason is that the complete graph shows the entire bibliography database. Thus the vertices with high degree will (most likely) be some standard reference (like "An Introduction to ..."), and we have no idea if this particular paper is useful for our new paper. The author reads the bibliography of this paper. Afterwards he copies some of the references this paper cites. For each of these new references, he recursively searches their references.

This could also explained on other social networks. For example a new student is at his first day at the university, but unfortunately he is the only person from his school at this university, so initially he has no one to work with. He befriends some older student, perhaps through a mentoring program. This older student introduces our new student to his the other participants of the mentoring program. Our new friend befriends some of the participants and is in return introduced to their friends. In this way our student is able to find several friends.

### 3.1.2 Formal Definiton of the Forest Fire Model

The Forest Fire Model [3] process vertices and edges in an alternating order. First a new vertex is inserted into the graph. Then we determine a subset of already connected vertices and form edges from these vertices to the newly formed vertex. This means, that when a new vertex is inserted, there are first all edges to existing vertices inserterd, before the next vertex is inserted into the graph. This means in particular, that when a vertex $x$ is inserted, then all edges which are inserted afterwards have $x$ as endpoint.

The insertion process of a single vertex consists of several "insertion depths". Procedure of insertion depth 0: In the following we assume that the graph already contains $n$ nodes. Now we add the vertex $v_{n+1}$ to the graph (initially $v_{n+1}$ is isolated). Choose an ambassador $w$ uniformly at random from the $n$ vertices which are already in the graph. Create an edge $(v_{n+1}, w)$. This is the equivalent to the choice of the first paper.

Procedure of insertion depth 1: Use a random number generator to get an number $k$ which is geometrically distributed with mean $\frac{p}{1-p}$. If $w$ has less than or exactly $k$ neighbors, we form an edge between $v$ and every neighbor of $w$. If $w$ has more than $k$ neighbors, we choose $x$ neighbors $\{w_1, ..., w_k\}$ of $w$ uniformly at random and form an edge between those chosen neighbors and $v_{n+1}$ and every neighbor of $w$ we just choose.

Procedure of insertion depth $m$. In the next insertion depth, we apply this recursive algorithm $k$ times for every $w_1, ..., w_k$ which was added at the previous depth. In this way we get an insertion tree with a variable number of children for each node within the tree, because the number of children depends on a random generator and the size of the neighborhood of the current vertex.

As on can see there are two abortion conditions: The first abortion condition occurs, if a vertex $w$ has no neighbors, which are not already connected to $v_{n+1}$. The second abortion condition occurs, if we are not allowed to add an further edge, because each of the $k$ random variables was a 0.

As a conclusion, the algorithmus only uses a single parameter $p$. Obviously for $p = 0$ we have an random tree creator, while for $p = 1$ we create the complete graph everytimes.

**Algorithm 1** Forest Fire Model (Recursive Version)

```
1:  procedure addNode(int ID)
2:      create new vertex v with given ID;
3:      Choose an ambassador w from G uniformly at random
4:      Create a new edge (v, w)
5:      Call recursion(v,w)
6:  procedure recursion( Vertex v, Vertex ambas)
7:      Generate a random number x from a geometrical distribution.
8:      Select x neighbors of ambas, which are not already connected with v,
9:      in the following called w_1, w_2, ..., w_x.
10:     If there are fewer than x possible neighbors, select all remaining neighbors.
11:     For every w_i: Create an edge (v, w_i) and call recursion(v,w_i)
```

## 3.2 Implementation

The Forest Fire Model is implemented with Java and Eclipse. Because our emphasis lies on fast graph creation and analysis, the implementation does not use a GUI, and the output is a data stream that is stored on the HDD. Changes of constants have to be entered by code, in order to to ensure that the constants are not changed accidentally, while a single instance is still running. Furthermore we did not use a templates for graph generation, such that we could implement a very primitive graph package which contains no unnecessary functions (no labeling of vertices/edges, vertices are consecutively numbered, and removal of vertices/edges is impossible). Generation and Analysis is done by the same program, because this enables us to store only the results of the analysis on the HDD, but no graphs. Our Projects is splited into three major parts: Part A: Generation of a graph. Part B: Analysis of a graph. Part C: Storage of the results.

### 3.2.1 Generator

As a first step, we convert the recursive definition of the Forest Fire Model into a iterative version, because this is much less time and memory consuming. Additionally we have to deal with a point which was only solved as pesudocode in the definition of the FFM: We need to ensure that no edge is inserted twice. For this, we maintain a list *burned*, which contains every neighbors of the new vertex $v$, so for a new candidate can easily checked if it already is neighbor of our new vertex. Furthermore we use two queues: The first queue currentQueue stores all neighbors which were inserted at the last recursion depth, and the second nextQueue contains the new vertices from the current depth.

Meanwhile, for every vertex we keep track of the insertion depth and degree of every vertex. In order to eliminate outliers, we create ten graphs with the same value for $p$.

**Algorithm 2** Forest Fire Model (Iterative Version)

```
1:  procedure ADDNODE(int ID)
2:      create new vertex v with given ID;
3:      Choose an ambassador w from G uniformly at random
4:      Create a new edge (v, w)
5:      Initialize currentQueue and nextQueue as empty list.
6:      burned.add(v);
7:      burned.add(w);
8:      currentQueue.add(w);
9:      for (depth = 1; depth <= maxDepth; i++) do
10:         //maxDepth either Integer.MaxValue or set by the user
11:         if currentQueue.isEmpty() then break; //In the last step there were no new nodes
12:         while !currentQueue.isempty() do
13:             current = queue.remove(0);
14:             Generate a random number x from a geometric distribution.
15:             Choose x neighbors from current which are not already in the list burned
16:             For each w_i: Create an edge (v, w_i) and insert w_i into queue2
            currentQueue = nextQueue.copy; nextQueue.clear;
```

### 3.2.2 Geometric distribution generator

What is a geometric distribution at all? Up to this point we only know the mean of the distribution. In this chapter, we are only interested in creating a random generator for the geometric distribution. Therefore explanation and proofs of the following statements are not in this chapter, but in the next chapter, which comprises our theoretical work, including proofs.

The geometric distribution is a discrete probability function. $P(X = k)$ only defined for $k \in N$. The probability mass function (pmf) of the geometric distribution is defined as: $P(X = k) = p_k = p^k \cdot (1 - p)$ The cumulative distribution function (cdf) of the geometric distribution is defined as: $P(X \leq k) = P_k = 1 - p^{k+1}$ This results in a expected value of $E[X] = \frac{p}{1-p}$ for the average of the geometric distribution.

We are not interested in a cryptological secure random generator, because we are not interested in the particular result of the random number generator, but we know that will have to use the generator very often. Thus it should be considerably fast. Therefore a simple pseudo random generator like java.util.random should suffice. However Java only offers a function uniform distribution, but no function for geometric distributed values. Therefore, we have to build agenerator by ourself. A naive approach would be to flip a skewed coin which lands on head with probability $p$, until the coin lands on tails the first time. However, this idea is not very efficient. The running time of the generator depends on the final result, because we call the generator of the coin flip more often for higher values of $p$. Especially for large $p$ this means the random generation alone takes more time. However, we need to create random values several times per instance of ADDNODE, therefore we need a faster random generator.

Therefore we use another approach, which has a better running time, by the use of Inverse transform sampling: For this, we generate a random number $x$ between 0 and 1, by a single call to java.util.random. In the following, we treat this value as the quantile of our geometric distribution. Thus we search for an $k$ with $P_k = x$. However, because $k$ must be an integer, it is unlikely to find an exact solution. Therefore, we relax the equation and just demand that $k$ satisfies $x < P_k$.

$$\begin{aligned} x &< 1 - p^{k+1} \\ x &\leq p^{k+1} \\ (k+1) \cdot \log(p) &\geq \log(x) \\ k + 1 &\geq \log(x)/\log(p) \\ k + 1 &= \lceil \log(x)/\log(p) \rceil \\ k &= \lceil \log(x)/\log(p) \rceil - 1 \end{aligned}$$

With this knowledge, we can develop a small function (see algorithm 3 below) which computes random values, which are geometric distributed, with the nice improvement, that we use only one call to java.util.random, regardless of the value of $p$.

### 3.2.3 Analysis Tools

Another part of the implemented algorithm has the assignment to analyze the created graphs. For this reason, we stop the Forest Fire Model at certain intervals, say after every 20 nodes and their

**Algorithm 3** Random Number generator

```
1: procedure GENERATEGEOMETRICRANDOM(DOUBLE PROP)
2:     double uniform = random.nextDouble();
3:     double log = Math.log(prop);
4:     int result = (int) Math.ceil(Math.log(uniform)/log)-1;
5:     return result;
```

corresponding edges are inserted into the graph. At this point we analyzse the current graph, before we restart adding vertices and edges into the graph. By default this procedure is repeated with 10 different graphs under otherwise constant circumstances (no change of the parameter).

We allow the user to select, if the maximum insertion depth should be bounded or de facto unlimited. If bounded depth is used, then the depth grows (first graph stops no later than depth 1, the second graph stops at pr before depth 2) If maximum insertion depth is not bounded then every graph uses `Integer.MaxValue` as upper bound. `Integer.MaxValue` can never be reached, because a vertex with a certain insertion depth $n$ as at least $n$ neighbors, because at least one neighbor is added per insertion depth. Therefore this trick is applicable, unless we want to use graphs with $2^{31} \approx 2$ billion vertices. Additionally the user is free to determine his desired frequency of analysis step (do you want to analyze every 50 nodes for small graphs, or every 1000 nodes for larger graphs?) and to define the amount of analysis. This is, because we computate the diameter, by using by the Floyd-Warshall-Algorithm, which has an expected running time of $\mathcal{O}(n^3)$ However, as a trade-off, if diameter computation is enabled, we also over the effective diameter, which is defined as the smallest value, such that 90% of all path are shorter or equal to this value. This is to eliminates possible outliers, very long paths, which are also called whiskers.

### 3.2.4 Storage of Results

Because the there programm may run for a long time, like several hours, on big graphs it would be inconvient, to just output the results into the command line, so that storage of results is in the responsibility of the user. Therefore the data is automatically stored on the Hard Disk, and the programm only summarizes the progress of the algorithm (How many graphs are completed, How large is the current graph), so that the user can observe the overall progress.

Inside the data of this project, there exists a folder "'DATA"', where all files, generated by the Forest Fire Model, are stored by default. When we run the programm, for every probability $p$, which we use to create graphs, the project creates a file with the following name: no of graphs + geom/cons + value of $p$ + ".csv". Our data is stored as a comma separated values (CSV) file, because this offers the following advantage: While this representation is very similar to .txt (can be read on a texteditor), this file can additionally be opened with many spreadsheets, like EXCEL

or LibreOffice. In the .file we store our data in the following order: Number of Vertices, Number of Edges, Alpha, Max depth, Average Depth, Max Degree, Min Degree, Avg Degree, Graph Time, Analysis Time, Diameter and Effective Diameter. Recall, that alpha $= \frac{\log(Edges)}{\log(Nodes)}$ was defined during our discussion about Densification Power Law. Under graph time, we store the time, needed to create the graph. Consequently analysis time, is the amount of time, we need to analyze our graph. All times are determined using `System.nanoTime()`.

After all analysis steps of the graphs are stored in the file, the following distributions are only computed once, and not at every analysis step. First the degree distribution of the ten graphs is appended to the data. Lastly the depth distribution of the ten graphs is appended.

## 3.3 Evaluation

After we have discussed the source code of the programm in every single detail, we can now progress to the results of the model generator:

### 3.3.1 Bounded recursion depth

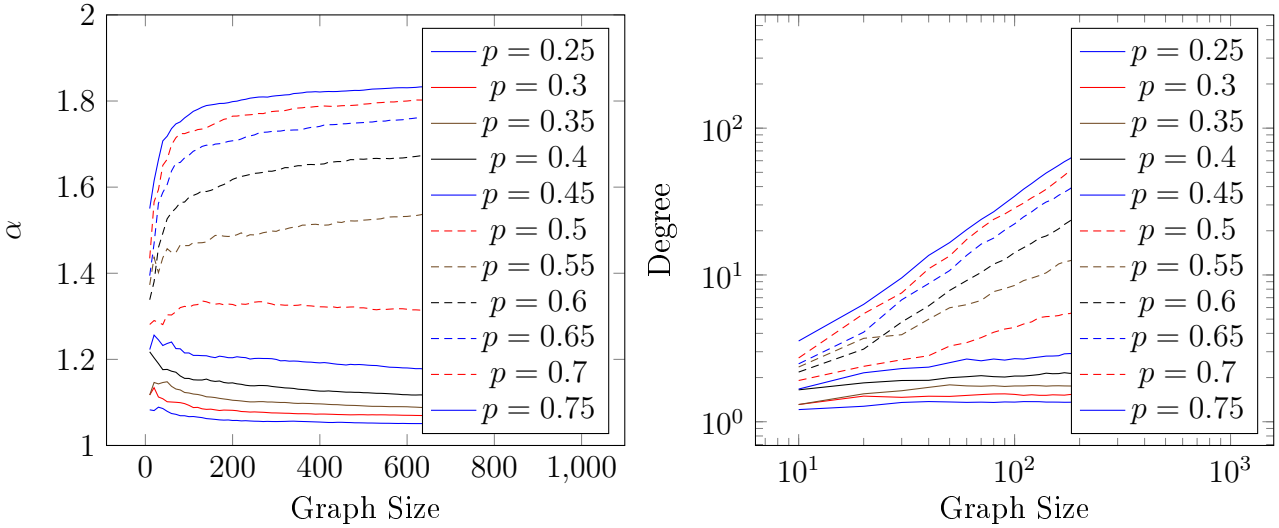
The $N$th graph has maximum allowed recursion depth $N$, which means that the last graph has a recursion depth of 10. This is, because the algorithm allows each graph one further insertion depth more than the previous graph. Note however, that this function is by no way meant to be used scientfic evidence, because of it s small sample size. Instead it should be understanded merely as a last check, if there are no hidden bugs (sanity check). For example, if a graph with maximum insertion depth $k$ has more edges than a graph with depth $k+1$, then there would be some issue with the underlying algorithm. However, this only gives us a first glance at the behaviour of the Forest Fire Model

### 3.3.2 Unbounded/Infinite recursion depth

This is our first real test, where we want to analyze our data: We create small graphs with up to 1000 vertices with $p = 0.25, 0.3, ..., 0.75$, in order to get an general overview over the behaviour of the Model. The results (see Figure 3.1) are, that for $p < 0.5$ both $\alpha$ and average degree seem to get constant, which is a paradoxon, because constant $\alpha > 1$ suggest densification, while constant average degree means sparsification.
Luckily, there is a simple explanation: We have just created graphs, which were too small, in order to determine which assumption is correct. For $p > 0.5$ only $\alpha$ gets constant, while the average degree keeps to grow continuosly, which means, that we are creating dense graphs.

As we expect, for $p > 0.5$ the algorithm takes much more time, because it takes more time to create dense graphs, therefore, in the rest of this chapter, we will focus on $p < 0.5$. Recall, that

Figure 3.1: Progression of $\alpha$ and average degree for graphs with up to 1000 vertices




the model should give better result for large graphs, because small graphs depends to much on chance (both the model as real world equivalents).

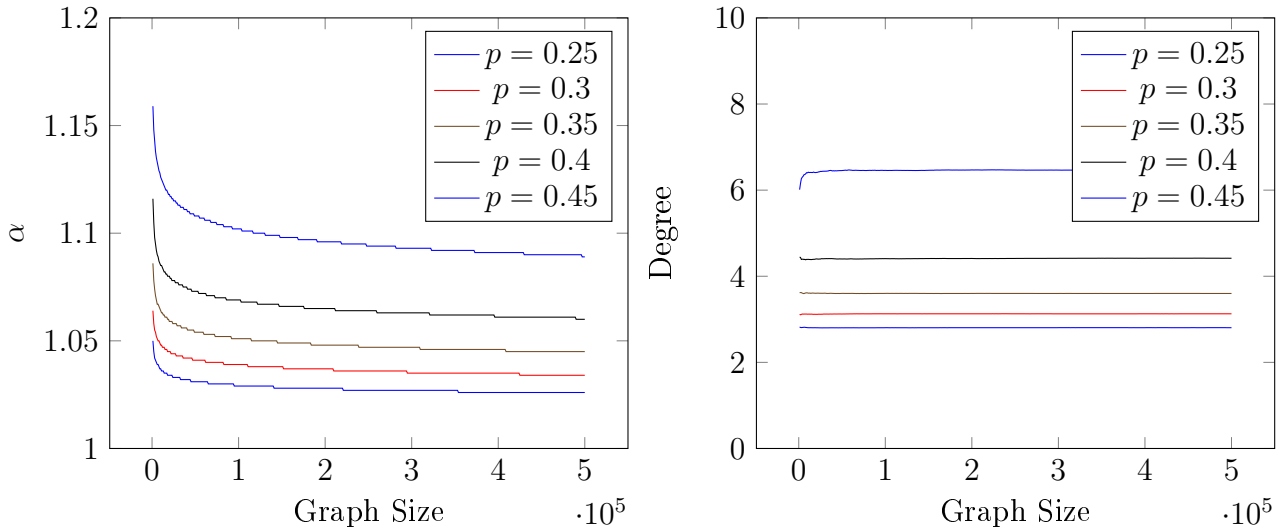
Now we examine much larger graphs (results are depicted in figure 3.2), with up to 500.000 vertices per graph. with $p = 0.25, 0.3, 0.35, 0.4, 0.45$.

With this data we can conclude, that $\alpha$ will asymptotically shrink to exactly 1, for any $p < 0.5$, which means that only the assumption about constant average degree was correct. This, in turn, means that average insertion depth is also constant, because it is impossible that the number of insertion depths grows, while the number of edges per insertion stay the constant. This means, that the number of edges in a graph depends linearly on the number of vertices, where the average degree depends solely on the probability of the generator (i.e our value chosen for $p$), but not on the size of the graph.

### 3.3.3 Restart graphs

One might argue, that using the average of ten graph is a very small sample size. Furthermore we have the problem, when we consider that the graphs of a have at an certain analysis step are heavily dependent in the graphs analyzed in the previous step, because we only added comparatively few vertices (for example, a graph is evaluated with 10.000 and 10.500 vertices). During a early preview of the result of the Forest Fire Model some people even questioned, if the diagrams were smoothed somehow.

Figure 3.2: Progression of $\alpha$ and average degree for large graphs with up to 500.000 vertices




Therefore the algorithms offers to following feature to the user: For this, we analyze ten graph of a given size (say 500 nodes), discard them, then create ten new graph of another size (say 1000 vertices) and so on. This means that we now create 10 graphs per analysis, while we previously only analyzed the same 10 graphs over and over again, at constant intervals. Unlike the previous implementation, the diagramms seem more coarse now (and less artificial smoothed), however this does not not change our findings from the previous section, the results are still very similar from one analysis step to the next.

## 3.4 Possible Variations

The algorithm described above creates undirected graphs, which can be used to model collaboration networks. The following possible variants are imaginable and easy to implement in future work: A directed graph, where loops are not allowed, which resembles for example a citation network. The algorithm would be almost the same as above, but we only have to define all created edges as directed (outgoing from the new vertex). However, this makes the computation of the diameter more complicated, because the graph will be (most likely) only weakly connected.

Another variant is the creation of directed graphs, where loops are possible, for example to model weblinks. For this, we simply call the algorithm for edges insertion twice per vertex. In the first instance, we only create outgoing links, in the second instance only incoming links for this vertex.

However, there is on restraint when using directed graphs: One has to consider, the model assumes that all outgoing edges of a new vertex are created at once (or at least before the next vertex en-

ters). Only the incoming degree distribution varies as the algorithm. On the other many webpages appear online, when they are still in progress (especially blogs), but the author wants feedback. This means the corresponding vertex will form additional outlinks long after its creation.

## 3.5 fixed fraction of neighbors

Leskovec et al. [3] suggested to simply use a certain fraction of the neighbors of the current ambassador, instead of a random distribution. Instead of the use of the geometric distribution, we determine the size of the fixed fraction of the neighborhood of an ambassador, and arbitrarily select a subset of neighbors Thus the number of new edges formed at a certain point of the insertion point depends on the existing edges from vertex $w$. A problem is here, however, when mathematical rounding is used, for small degree rounding up is very uncommon. For example, assume $p = 0.1$, then a vertex needs already 5 neighbors before we expect to round up for the first time. Therefore we use "random" rounding: A coin flip decides if we round up or down. Thus the first decimal place is irrelevant. The following code shows, how we determine at a certain insertion depth, how many new edges are inserted. We only have to ensure that Algorithm 2 calls this new function instead of generating a new geometric distributed number in line 14.

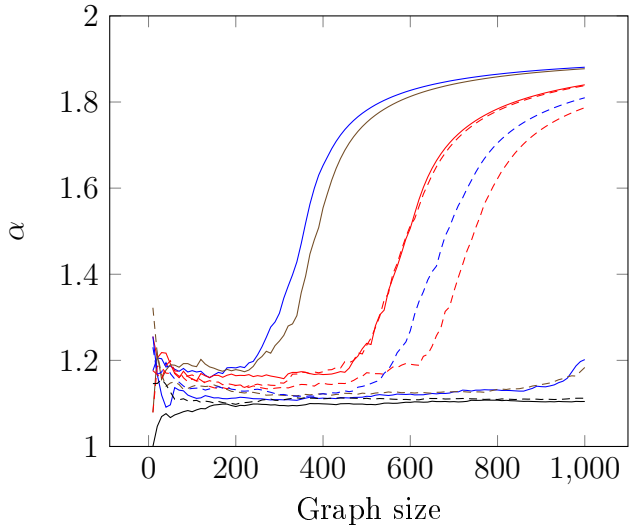
**Algorithm 4** fixed fraction

```
1: procedure ADDNODE(INT ID)
2:     x ← fraction * |neighbors|
3:     if random.nextDouble ≤ 0.5 then
4:         x++;
```

However we have to recognize (see Figure 3.3), that for every $p > 0$ there seem to exist an corresponding graph-size, at which the graph becomes very clique-ish. Even for $p = 0.045$, a graph with $|V| = 500$ might contain up to 10.000 edges, while a graph of the same size, generated with geometric distribution and $p = 0.25$, has about 800 edges. Even the minimum degree of the graph grows rapidly, which means that the results are no longer heavy-tailed, but instead even short-tailed.

Consider for example the 10 graphs, which are depicted in Figure 3.3, which were all created with the same fraction $p = 0.045$. As one can see the cutoff, where the graph switches from very sparse ($\alpha \approx 1$) to very dense ($\alpha \approx 2$), is different for each sample graph. While 2 graphs have their cutoff at $n \approx 300$ vertices, several other have their cutoff at $n \approx 600$, while the resmaining graphs, do not exhibit the cut-off at all. Thus the use of constant fraction is not recommended, because it not only creates random graphs, but the graph properties diverge, despite the variables are not changed. Thus it is almost impossible to find any similarities between two graphs, which is a rather bad feature for a model.

Figure 3.3: 10 graphs, created with "fixed fraction" variant and $p = 0.045$




A possible reason is, that for every $p$ there exists a possibility that a single fire "burns" a large fraction of the graph, in the process ensuring that the burned vertices will become a large degree. Subsequent fires will be more likely to follow this path, visiting more or less the same vertices, as soon as one vertex from the first fire is reached. Thus the introduction of a new vertex relies too much on previous occurences in the same graph. Because of this, the geometric model generator is much faster, than the constant fraction generator, and gives better (i.e. more reliable) results.

# Chapter 4

# Combinatoric Analysis

In the previous chapter we have seen that for every $p < 0.5$ there seems to exist an upper bound for the number of edges. Furthermore we could see, that there is a major change in the power-law distribution of the resulting graphs at soon as $p \geq 0.5$. Thus we are interest, if it is possible to mathematically prove an upper bound for the average insertion depth and the number of added edges per vertex. In this chapter, we use the assumption the number of edges only depends on the geometric distribution. This means that the search tree is never aborted because there are not enough possible neighbors, but only if the geometric distribution gives out a 0. This assumption becomes increasingly unrealistic for large $p$, but should give fairly good results for small values of $p$. We use a small program (see algorithm 5 below) that only tests the probabilities, but does not create an actual graph.

As one might notice, this is almost the same algorithmus like our iterative version of the Forest Fire Model. The major difference is, that we do not actually try to create a graph, but instead merely sum over all results from the random generator. We let this small program run by increasing $p$ in steps of 0.01 from $p = 0$ onwards and using a sample size of 1.000.000 tests per value of $p$. This tester is much faster than the Forest Fire Model, because we are not creating vertices and edges here, but instead, we basically draw random number after random number and compute their sum, which allows much larger sample sizes. Whenever 1.000.000 calls are computed, we store the average "added edges" and "insertion depth" for the current value of $p$. A sample of the result of this program are presented in figure 4.1.

There are no results for $p \geq 0.5$, because the at least some of the calls to the generator test never stop their iteration, but seem to literally run forever. As we expect, $avgNodes$ and $avgDepth$ are monotonically increasing, when we increase $p$. We observe, that $avgNodes$ and $avgEdges$ initally depend linearly on $p$. The nearer $p$ gets towards 0.5, the steeper get both curves.

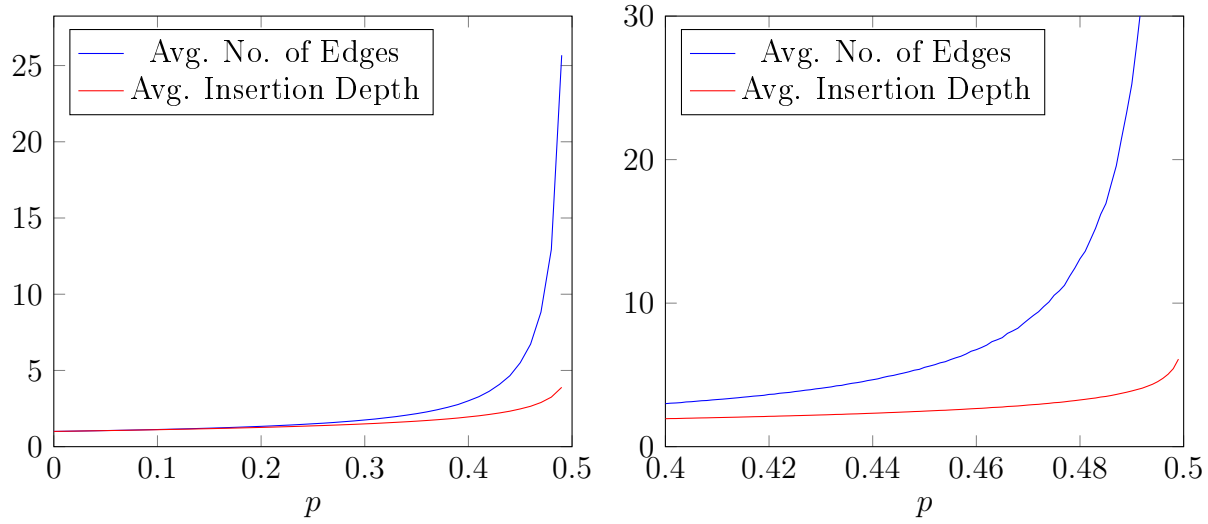
This supports our results from the previous section, namely that the behaviour of the graph changes drastically for $p \geq 0.5$ On the other hand the geometric distribution says, that even

**Algorithm 5** Generator Tester

**Require:** Edgesum and Depthsum are static counters, to keep track of the overall inserted edges and depth reached.

```
1:  procedure SINGLERUN(DOUBLE PROB)
2:      NewEdges ← 1;
3:      Edgesum + +;
4:      Depthsum + +;
5:      nextEdges = 0;
6:      while NewEdges! = 0 do
7:          while NewEdges! = 0 do
8:              NewEdges − −;
9:              random = generateGeometricRandom(prob);
10:             nextEdges+ = random;
11:             Edgesum+ = random;
12:         if nextEdges > 0 then
13:             NewEdges ← nextNodes;
14:             nextnodes ← 0;
15:             Depthsum + +;
```

Figure 4.1: Results of the Generator Tester



$p = 0.5$ means, that in 50% of the new vertex is only connected to the ambassador, but to no other vertex. This means that the insertion depth of the corresponding vertex is 1. For the further insertion depths we can expect similiar behavior. Common sense, says, that it should be more likely to stop than to continue. The question, is can we still explain why the number of edges seems to explode at this point?

The insertion of a single vertex in to the graph can be compared with the Galton-Watson-Process [6]. The Galton-Watson Process orginates from genealogy: It was developed to estimate the survival probabilites of certain surnames. In our case, we want to determine the probability that our insertion is still alive at a certain depth. A necessary condition of the Galton-Watson Process is, that all edges are inserted with some independent and identically distributed (iid) random generator. This also means that we are not allowed to change the probability $p$ at any point of the algorithm. On other words, the result of the 100th generated number does not depend on the previous draws and the distribution is not changed, for example from geometric to poisson distribution. Also not allowed is a change of the parameter $p$ at a certain insertion depth. Thus the comparison to the Galton-Watson Process makes even more sense in our case, than the original application, where the average number of born kids varies over times. On the other hand we can not compare our fixed fraction variant, because here the probabilites are not independent. If a large fire just has happend other fire are more likely to follow this path.

## 4.1 Geometric distribution

The Forest Fire Model uses a geometric distribution to sample the number of edges in a single recursion step. The name comes from similiarites to the geometric sequence $(a, ap, ap^2, ...)$. In our case, we simply have to define $a := 1 - p$. Beware confusion: Sometimes the resulting probability is defined as chance that the first success happens at step $k$. In our case it would be awkward to call the insertion of an edge a failure. We use the not so frequently used opposite definition: We count the number of successes, until the first failure occurs. The geometric distribution is a discrete, heavy tailed distribution, which is explained as follows..

$$p_k = P(X = k) = \begin{cases} 1 - p, \text{ if } k = 0 \text{ and} \\ p \cdot P(X = k - 1), \text{ otherwise.} \end{cases}$$

$$\sum_{k=0}^{\infty} p_k = 1$$

Recall it makes sense that the used probability is heavy tailed, because we want the overall distribution to be heavy tailed. A discrete distribution defines $P(X = k)$ only for $k \in N$. A heavy-tailed distribution, demands $P(X = k) > 0$ for all $x \in N$ and $P(X = k) > P(X = k + 1)$. As a conclusion, in a single instance of the geometric distribution, we are allowed to add further edges with propability $p$ until we reach the first failure.

**Definition 4.1.** *A geometric function has the following probability mass function:*

$$P(X = k) = p^k \cdot (1 - p) \tag{4.1}$$

Before we continue our research about the geometric distribution, let us have a look back towards its namesake, the geometric series. It is recommended to use the following lemma:

**Lemma 4.2.** *Use the following formula for the geometric series*

$$S(a) = \sum_{k=0}^{\infty} a^k = \frac{1}{1-a} \tag{4.2}$$

*Proof.*

The geometric sum is defined by:

$$S(a) = \sum_{k=0}^{\infty} a^k$$

Now we multiply the equation above by $a$

$$\begin{aligned} a \cdot S(a) &= a \sum_{k=0}^{\infty} a^k \\ &= \sum_{k=1}^{\infty} a^k \end{aligned}$$

Now subtract the last equation from the previous one

$$\begin{aligned} S(a) - a \cdot S(a) &= \sum_{k=0}^{\infty} a^k - \sum_{k=1}^{\infty} a^k \\ &= (1 + a + a^2 + a^3 ...) - (a + a^2 + a^3 + ...) \end{aligned}$$

Only the 1 appears once in the subtraction, all other values apper twice and can thus be eliminated.

$$S(a) - a \cdot S(a) = 1$$

Factorize $S(a)$ on the left hand side of the equation

$$(1 - a)S(a) = 1$$

Divide by $1 - a$

$$S(a) = \frac{1}{1-a}$$

Further explanation: This only holds for $|a| < 1$. Otherwise the series diverges. □

In our case it suffices to assume that $0 \leq a < 1$.

**Lemma 4.3.** *Use the following formula fto compute the sum of the n first elements of the geometric series*

$$S_n(a) = \sum_{k=0}^{n-1} a^k = \frac{1-a^n}{1-a} \tag{4.3}$$

*Proof.* Basically, we reuse, the proof from Lemma 4.2 here

$$S_n(a) = \sum_{k=0}^{n-1} a^k$$

Now we multiply the equation above by $a$:

$$\begin{aligned} a \cdot S_n(a) &= a \sum_{k=0}^{n-1} a^k \\ &= \sum_{k=0}^{n} a^{k+1} \end{aligned}$$

Now subtract the last equation from the first one:

$$\begin{aligned} S_n(a) - a \cdot S_n(a) &= \sum_{k=0}^{n-1} a^k - \sum_{k=0}^{n-1} a^{k+1} \\ &= (1 + a + a^2 + ... + a^{n-1}) - (a + a^2 + a^3 + ... + a^{n-1} + a^n) \end{aligned}$$

Only the 1 and $a^n$ appears once in the subtraction, all other values apper twice and can thus be eliminated:

$$S_n(a) - a \cdot S_n(a) = 1 - a^n$$

Factorize $S_n(a)$ on the left half of the equation:

$$(1-a)S_n(a) = 1 - a^n$$

Divide by $1 - a$:

$$S_n(a) = \frac{1-a^n}{1-a}$$

It has to be noted, this result only holds for $|a| \neq 1$, because we have a division by 0 iff $|a| = 1$. □

Now we can derive the expected value $\mu$ and the cumulative distribution function.

**Lemma 4.4.** *The mean or average value of the geometric distribution is* $\mu = E[X] = \frac{p}{1-p}$

*Proof.*

Definition of the expected value:

$$\begin{aligned}\mu &= \sum_{k=0}^{\infty} P(X = k) \cdot k \\ &= \sum_{k=0}^{\infty} p^k \cdot (1-p) \cdot k\end{aligned}$$

Eliminate the bracket and split the sum into two parts:

$$\mu = \sum_{k=0}^{\infty} p^k \cdot k - \sum_{k=0}^{\infty} p^k \cdot p \cdot k$$

First element of the left sum is $p^0 \cdot 0 = 0$, therefore we can start counting from $k = 1$ in the left sum:

$$\mu = \sum_{k=1}^{\infty} p^k \cdot k - \sum_{k=0}^{\infty} p^k \cdot p \cdot k$$

Substitute $k$ such that the sum starts with $k = 0$ again:

$$\mu = \sum_{k=0}^{\infty} (k+1) p^{k+1} - \sum_{k=0}^{\infty} p^{k+1} \cdot k$$

Concatenate the two sums again, and simplify as most as possible:

$$\begin{aligned}\mu &= \sum_{k=0}^{\infty} (k+1-k) p^{k+1} \\ &= \sum_{k=0}^{\infty} p^{k+1}\end{aligned}$$

Factorize $p$ from the sum:

$$\mu = p \sum_{k=0}^{\infty} p^k$$

According to the geometric series (Lemma 4.2), the sum is equal to $\frac{1}{1-p}$

$$\mu = \frac{p}{1-p}$$

□

This means that if $p = 0.5$, we can expect that the mean of inserted edges per application of

the geometric distribution will be converging towards $\frac{0.5}{1-0.5} = 1$, while the mean will only be 0.33 expected edges, if $p = 0.25$ is used. Furthermore the distribution is heavy tailed, which means that smaller results like 0 or 1 are more likely than than larger values.
Note that this expected value only holds for the average of the random values. This does not mean that we can expect that on average $\frac{p}{1-p}$ edges are inserted at the first insertion depth of the Forest Fre Model, because in case that the generator gives a high number $k$, there is the possibility that the currently choosen ambassador does not have enough neighbors to add exactly $k$ edges.

**Lemma 4.5.** *A geometric distribution has the following cumulative distribution function(CDF):*

$$P(X \leq x) = 1 - p^{x+1} \tag{4.4}$$

*Proof.*

Definition of the cumulative distribution:

$$\begin{aligned} P(X \leq x) &= \sum_{k=0}^{x} P(X = k) \\ &= \sum_{k=0}^{x} p(1-p)^k \end{aligned}$$

Factorize $p$ from the sum

$$= p \sum_{k=0}^{x} (1-p)^k$$

The sum is identic to $S_{x+1}(1-p)$ to the geometric sum with $a = 1 - p$ and $n = x + 1$

$$\begin{aligned} &= p \frac{1 - p^{x+1}}{1 - (1-p)} \\ &= 1 - p^{x+1} \end{aligned}$$

□

With this we can explain everything that happens at the first insertion step of the algorithm.

## 4.2 Multiple Iterations of geometric Distributions

Up to this point we only have considered single applications of the geometric distribution. We are now able to explain everything that happens at insertion depth 0 and 1. In theory, it is fairly easy, to compute the sum of $m$ applications of the geometric series, becuae this is the negative binomial distribution with parameters $p$ and $m$. However, we have the problem, that the number of calls to the geometric distribution is not constant (as required by the negative binomial distribution), but depends on the result of the previous calls of the geometric distribution.

### 4.2.1 A Simple Recurrence Relation

A recurrence relation is a recursive relation.

**Lemma 4.6.** *Let $Z$ be the average number of edges added by a single call of the Forest Fire Model (i.e. in new vertex arrives).*

$$Z = \frac{1-p}{1-2\cdot p} \tag{4.5}$$

*Proof.* Recall, the recursive definition of the Forest Fire Model (see chapter 3): Add the edge to the ambassador, then call $k$ new instance of the algorithm at the next depth, where $k$ is given by a geometric distrution generator.
In other words, at any insertion depth, we first add a new edge and then progress on each new ambassador. From this it is very straightforward to derive a recurrence relation for the overall number of inserted edges:

$$Z = 1 + \sum_{k=0}^{\infty} P(X=k)\cdot k \cdot Z$$

The sum adds all branching possibilities, including their probabilites

$$Z = 1 + \sum_{k=0}^{\infty} (1-p)\cdot p^k \cdot k \cdot Z$$

$1-p$ and $Z$ are constants and can thus be factorized from the sum

$$Z = 1 + (1-p)\cdot Z \cdot \sum_{k=0}^{\infty} p^k \cdot k$$

The sum is equal to the definition of the expected value of the geometric distribution

$$\begin{aligned}
Z &= 1 + Z\cdot\frac{p}{1-p} \\
1 &= Z - Z\cdot\frac{p}{1-p} \\
1 &= Z\cdot\frac{1-p}{1-p} - Z\cdot\frac{p}{1-p} \\
1 &= Z\cdot\frac{1-p-p}{1-p} = Z\cdot\frac{1-2\cdot p}{1-p} \\
Z &= \frac{1-p}{1-2\cdot p}
\end{aligned}$$

□

Example: $p = 0.25$ gives $Z = \frac{1-0.25}{1-2\cdot 0.25} = \frac{0.75}{0.5} = 1.5$ $p = 0.4$ gives $Z = \frac{1-0.4}{1-2\cdot 0.4} = \frac{0.6}{0.2} = 3$. This gives the same results as the generator test. However, While this Lemma is able to detect the

densification of the graphs, created with $p \geq 0.5$, by creating unrealistic (i.e. negative) values, there are still two open questions, which can not be answered by this recurrence relation: The first question is, how many edges are inserted at a certain recursion depth. If we can show that, a large fraction (say 95 or 99 per cent) of the edges are inserted at or before a certain insertion depth, we can stop the recursion at this depth without missing to many edges and save precious computation time. Additionally we might find an upper bound for the average insertion depth.

The other question, is how the model behaves, if $p \geq 0.5$. The formula from Lemma (4.6) gives negative values for $p \geq 0.5$, but this clearly does not makes sense, because the algorithm does not allow removal of edges at any point. In order to derive a recurrence relation for the insertion depth, we need another recurrence, which enables us to directly estimate the number of edges inserted at a certain insertion depth.

### 4.2.2 Advanced Recurrence Relation for Insertion Depth:

Now we develop an advanced recurrence relation, which allows us to estimate the number of edges inserted at a certain insertion depth.
We define $Z_n$ as number of edges inserted at depth $n$, and
$X$ as random variable sampled from geometric distribution. Next we can directly estimate the number of edges at a certain insertion depth in dependence of the previous insertion depth

$$Z_n = \begin{cases} 1 & \text{if } n = 0 \\ \sum_{k=0}^{Z_n} X_{n,k} & \text{if } n > 0 \end{cases} \tag{4.6}$$

Note that $Z_n = 0 \Rightarrow Z_{n+1} = 0$ holds. which makes sense, because the algorithm cannot restart out of nothing. Recall that the Galton-Watson Process has the condition that all $X_{n,k}$ are independently distributed. This means that we cannot predict the result of $X_{n,k}$, even if we know $Z_{n+1}$, because they are absolutely random, but on the other hand the expected value of $X_{n,k}$ is $\mu$ everytimes, and thus does not depend on either $n$ or $k$. Therefore, we call $E[Z_{n+1}|Z_n]$ the expectation value of $Z_{n+1}$ under the condition that $Z_n$ is fixed but unknown.

$$E[Z_{n+1}|Z_n] = \mu \cdot Z_n \tag{4.7}$$

Of course, we can not compute $Z_n$, so we substitute it by its expected value $E[Z_n]$. Thus we get $E[Z_0] = 1$ and $E[Z_{n+1}] = \mu \cdot E[Z_n]$.
With this we can define:

$$E[Z_n] = \begin{cases} 1 & \text{if } n = 0 \\ \mu^n & \text{if } n > 0 \end{cases} \tag{4.8}$$

### 4.2.3 Probablity generating function of geometric distribution

The following sections will use the probability generating function (PGF) of the geometric function. Therefore we will first present the properties of the PGF and derive the PGF for the geometric distribution.

The PGF is a power series representation of the probability mass function, which is defined as:

**Definition 4.7.** $G(z) = \sum_{k=0}^{\infty} P(X = k) \cdot z^k$.

A nice property of a probability generating function is, that we can derive the individiual probabilities from the PGF, by computing the derivatives: $G(0) = \sum_{k=0}^{\infty} P(X = k)0^k = P(X = 0)$

$G'(0) = \sum_{k=1}^{\infty} P(X = k)k \cdot 0^{k-1} = P(X = 1)$

$G''(0) = \sum_{k=1}^{\infty} P(X = k)k \cdot (k-1) \cdot 0^{k-2} = 2 \cdot P(X = 1)$

$G^{(k)}(0) = k! \cdot P(X = 1)$

While the derivations of a probability generating function are rather difficult to compute, it is rather easy to compute the PGF for any probability distribution. Now we discuss the PGF of the geometric distribution:

$$\begin{aligned} G(z) &= \sum_{k=0}^{\infty} P(X = k) \cdot z^k \\ &= \sum_{k=0}^{\infty} p^k \cdot (1-p) \cdot z^k \\ &= (1-p) \sum_{k=0}^{\infty} p^k \cdot z^k \\ &= (1-p) \sum_{k=0}^{\infty} (p \cdot z)^k \\ &= \frac{1-p}{1-p \cdot z} \end{aligned}$$

### 4.2.4 Recurrence Relation for the extinction prabability

In the following section, we will use the following naming convention:

**Definition 4.8.** $p_n^{stop}$*: Stopp Probability: Probability that the process stops at exactly depth* $n$*.*
$p_n^{ext}$*: Extinction Probability: Probability the the process stops at depth* $n$ *or earlier.*

*$p^{ext}$: The probability that the process stops at some point.*

Notably, $p^{ext} < 1$, means that the process of edge incertion might continue forever, or if there are no remaining neighbors left. In the first step, we are only interested in the cumulative distribution function, not in the probability to stop at exactly depth $n$. Expected behavior of $p_{ext}$ and $p_n^{ext}$: For every $p$ there exists an $p^{ext}$ with the following conditions:
$p_n^{ext}$ is monotonically increasing.
$p^{ext} = \lim_{n\to\infty} p_n^{ext}$
$p_1^{ext} \leq p^{ext} \leq 1$

We now work from the back to the front, which means, that we first try to find a closed formula for $p^{ext}$ and for $p_n^{ext}$. This in turn, will enable us to also compute each $p_n^{stop}$ directly.

**Lemma 4.9.**

$$p_{n+1}^{ext} = \begin{cases} 0 & n = 0 \\ G(p_n^{ext}) & otherwise \end{cases} \tag{4.9}$$

*Proof.* We prove this statement by determining the maximum size of each subtree: Because the Forest Fire Model can be interpreted as a Galton-Watson Process, the subtrees exhibit self-similarity. $p^{ext}$ is equal to the probability that the insertion tree is finite: $P(|T| < \infty)$. $p_n^{ext}$ is equal to the probability that the insertion tree is depth of at most $n+1$. Therefore for every child of the root, the corresponding subtree has depth at most $n$.

$$\begin{aligned} p_{n+1}^{ext} &= \sum_{k=0}^{\infty} (p_n^{ext})^k p_k \\ &= \sum_{k=0}^{\infty} (p_n^{ext})^k p^k (1-p) \\ &= (1-p) \sum_{k=0}^{\infty} (p_n^{ext})^k p^k \\ &= (1-p) \sum_{k=0}^{\infty} (p_n^{ext} \cdot p)^k \\ &= \frac{1-p}{1-p \cdot p_n^{ext}} \\ &= G(p_n^{ext}) \end{aligned}$$

□

This recurrence relation can be used to compute the probability that the recursion stops at all:

Per definition of $p_{ext}$ holds:

$$\begin{aligned} p_{ext} &= \lim_{n\to\infty} G^n(p^{ext}) \\ p_{ext} &= \lim_{n\to\infty} G(G^{n-1}(p^{ext})) \\ p_{ext} &= G(p_{ext}) \text{ since } \lim_{n\to\infty} G^{n-1}(p^{ext}) = \lim_{n\to\infty} G^n(p^{ext}) = p_{ext} \end{aligned}$$

This means we are in search of the fixpoint of $z = G(z)$, Obvious for $p_{ext} = 1 \Rightarrow 1 = p_{ext} = G(1) = \frac{1-p}{1-p} = 1$ holds for every $p < 1$, thus 1 is a fixpoint regardless of $p$. The interesting question is, if 1 is the only fixpoint, or if there might appear a second fixpoint of $G(z)$.

Definition of $G(p_{ext})$:

$$p_{ext} = \frac{1-p}{1-p \cdot p_{ext}}$$

While this may sound awkward at first, we solve this equation for $p$ first, and not for $p_{ext}$

$$\begin{aligned} 1-p &= p_{ext} \cdot (1-p \cdot p_{ext}) \\ 1-p &= p_{ext} - p \cdot p_{ext}^2 \\ p \cdot p_{ext}^2 - p &= p_{ext} - 1 \\ p \cdot (p_{ext}^2 - 1) &= p_{ext} - 1 \\ p &= \frac{p_{ext} - 1}{p_{ext}^2 - 1} \end{aligned}$$

Use binomial formula $a^2 - b^2 = (a+b)(a-b)$

$$\begin{aligned} p &= \frac{p_{ext} - 1}{(p_{ext}+1)(p_{ext}-1)} \\ p &= \frac{1}{p_{ext}+1} \end{aligned}$$

Now as we only have one remaing instance of $p_{ext}$ we can solve for $p_{ext}$

$$\begin{aligned} (p_{ext}+1)p &= 1 \\ p_{ext}+1 &= \frac{1}{p} \\ p_{ext} &= \frac{1}{p} - 1 = \frac{1}{\mu} \end{aligned}$$

This means that the function $G(z)$ has exactly two fixpoints 1 and $\frac{1}{\mu}$. However, we know that $p < 0.5$ results in $\mu < 1$ and $\frac{1}{\mu} \geq 1$. We are only interested in the smallest fixpoint $\min\{1, \frac{1}{\mu}\}$, because a extinction probability of more than 100% clearly makes no sense. Therefore, if $p > 0.5$

there is a possibility that the generation goes on forever.

With this we can now derive a direct formula of the value for $p_{n+1}^{ext}$.

**Lemma 4.10.** $p_n^{ext} = \frac{1-p}{1-p \cdot p_{n-1}^{ext}} = G^n(0) = \frac{1-\mu^n}{1-\mu^{n+1}}$

*Proof.* The formula above is a first order rational difference equation, which are also called Riccati difference equation. Unfortunately, for this kind of formula it is very hard to find a direct function, because the previous element occurs in the denominator of the equation. Therefore we introduce a linear recurrence relation, for which it is much simpler to find a closed formula, because this relation is linear:

$$x_n = \begin{cases} -\mu, & \text{iff } n = 0, \\ \mu \cdot x_{n-1} - \mu, & \text{otherwise.} \end{cases}$$

For a recurrence relation it is fairly easy to find a closed formula to compute every value directly:

$$\begin{aligned} x_0 &= -\mu \\ x_1 &= \mu \cdot x_0 - \mu = -\mu^2 - \mu \\ x_2 &= \mu \cdot x_1 - \mu = -\mu(\mu^2 - \mu) - \mu = -\mu^3 - \mu^2 - \mu \\ x_n &= -\sum_{k=1}^{n+1} \mu^k = -(\sum_{k=0}^{n+1} \mu^k - 1) \end{aligned}$$

This is the negative version of the geometric series, therefore we can compute every $x_n$ directly with Lemma 4.3

$$\begin{aligned} x_n &= -(\frac{1-\mu^{n+2}}{1-\mu} - 1) \\ &= -(\frac{1-\mu^{n+2}}{1-\mu} - \frac{1-mu}{1-mu}) \\ &= -(\frac{\mu-\mu^{n+2}}{1-\mu}) \\ &= \frac{\mu-\mu^{n+2}}{\mu-1} \end{aligned}$$

Now we claim that there is a connection between $x_n$ and $p_n$, in other words, if we know a certain $x_n$, then we can directly compute the corresponding value for $p_n$.

**Lemma 4.11.** *The connection between $p_n$ and $x_n$ is as follows: $p_n = \frac{\mu+x_n}{\mu \cdot x_n}$ and $x_n = \frac{\mu}{\mu \cdot p_n - 1}$*

*Proof.* The validity of this claim can be proven by induction:

$$\begin{aligned} p_0 &= \frac{\mu + x_0}{\mu \cdot x_0} \\ &= \frac{0}{-\mu^2} = 0 \end{aligned}$$

For every $n > 0$ we show that $p_n = f(x_n) = g(x_{n-1}) = G(p_{n-1})$:

$$p_n = \frac{\mu + x_n}{\mu \cdot x_n}$$

Now we substitute both instances of $p_n$ by $\mu \cdot x_{n-1} - \mu$

$$\begin{aligned} p_n &= \frac{\mu + \mu \cdot x_{n-1} - \mu}{\mu \cdot (\mu \cdot x_{n-1} - \mu)} \\ &= \frac{\mu \cdot x_{n-1}}{\mu^2 \cdot (x_{n-1} - 1)} \\ &= \frac{x_{n-1}}{\mu \cdot (x_{n-1} - 1)} \end{aligned}$$

Now we substitute both instances of $x_{n-1}$ by $\frac{\mu}{\mu \cdot (p_{n-1} - 1)}$

$$p_n = \frac{\frac{\mu}{\mu \cdot p_{n-1} - 1}}{\mu \cdot (\frac{\mu}{\mu \cdot p_{n-1} - 1} - 1)}$$

Now we change the bottom of the large fraction, so that it only consists of a single denominator

$$\begin{aligned} p_n &= \frac{\frac{\mu}{\mu \cdot p_{n-1} - 1}}{\mu \cdot (\frac{\mu}{\mu \cdot p_{n-1} - 1} - \frac{\mu \cdot p_{n-1} - 1}{\mu \cdot p_{n-1} - 1})} \\ &= \frac{\frac{\mu}{\mu \cdot p_{n-1} - 1}}{\mu \cdot (\frac{\mu - \mu \cdot p_{n-1} + 1}{\mu \cdot p_{n-1} - 1})} \\ &= \frac{\frac{\mu}{\mu \cdot p_{n-1} - 1}}{\frac{\mu^2 - \mu^2 \cdot p_{n-1} + \mu}{\mu \cdot p_{n-1} - 1}} \end{aligned}$$

Now both fractions have the same denominator

$$\begin{aligned} p_n &= \frac{\mu}{\mu^2 - \mu^2 \cdot p_{n-1} + \mu} \\ &= \frac{1}{\mu - \mu \cdot p_{n-1} + 1} \\ &= \frac{1}{\frac{p}{1-p} - \frac{p}{1-p} \cdot p_{n-1} + \frac{1-p}{1-p}} \\ &= \frac{1}{\frac{p - p \cdot p_{n-1} + (1-p)}{1-p}} \\ &= \frac{1-p}{p - p \cdot p_{n-1} + (1-p)} \\ &= \frac{1-p}{1 - p \cdot p_{n-1}} = G(p_{n+1}) \end{aligned}$$

Clearly this is the same reccurence relation for $p_n$, as we defined in Lemma 4.8. □

Up to this point we have shown, that there exists a $p_n$ and $x_n$ and that every $x_n$ can be cumputed directly. Therefore we can use this information to find a closed formula for $p_n$.

$$\begin{aligned} p_n &= \frac{\mu + x_n}{\mu \cdot x_n} \\ &= \frac{\mu + \frac{\mu - \mu^{n+2}}{\mu - 1}}{\mu \cdot \frac{\mu - \mu^{n+2}}{\mu - 1}} \\ &= \frac{\frac{\mu^2 - \mu^{n+2}}{\mu - 1}}{\frac{\mu^2 - \mu^{n+3}}{\mu - 1}} \\ &= \frac{\mu^2 - \mu^{n+2}}{\mu^2 - \mu^{n+3}} \\ &= \frac{\mu^2 \cdot (1 - \mu^n)}{\mu^2 \cdot (1 - \mu^{n+1})} \\ &= \frac{1 - \mu^n}{1 - \mu^{n+1}} \end{aligned}$$

□

With this knowledge, it is now very easy to compute the probability that the algorithm stops at exactly depth $n$, by subtracting $p_n^{stop} = p_{n+1}^{ext} - p_n^{ext} = \frac{1-\mu(n+1}{1-\mu^{n+2}} - \frac{1-\mu^n}{1-\mu^{n+1}}$

With this we have shown, that for every $p < 1$ it is possible, to compute the number of edges at each insertion depth, the probability that the algorithm stops at a certain insertion depth, and the probability that the algorithm stops at all (or is stopped automatically because there are no new borders. In theory it would be possible to compute the average insertion depth from $p_n^{stop}$, but this is rather unadvisable because the formula would become rather complicated. It makes more sense to compilate a table of several stop probabilities and derive the average insertion depth by hand.

# Chapter 5

# Conclusion

The main part of this Bachelor Thesis is the definition and analysis of a simple model which crates graph, that resemble network graphs. Because it is completely impossible the predict the neighborhood of a random vertex, we only demand, that the general graph should resemble a network graph. With this model want to answer open questions, like is a network grasph sparse or dense, if it reasonable to discuss "$k$ degrees of separation" on every abitrary network graph, for example an citation network.

We start with a short analysis of the Erdős-Rényi Model. As expected we see that, despite the imprecise boundaries of network graphs, there are still somereuquirements for network graph model. Despite the absence of a precise definition for network graphs, we can still show, that there are several objective measures, which indicate, how we can distinguish between arbitrary graphs and network graphs. These include espacially (ja was den eigentlich)

The Forest Fire Model is a branching process, which means that we pick an arbitrary vertex of the existing graph, select a subset of its neighbors, add further edges and select new subset of neighbors, until we run out of neighbors. Interestingly, the Forest Fire Model, despite its simplicity, resembles the way how a real network is created fairly good, because we can explain its rules from a sociological point of view.

In the analysis of the Forest Fire Model, we notice that there, is a sharp transition between very sparse and very dense graphs at around $p = 0.5$. This corresponds to very well to reality, where some graphs might exhibit densification, especially where forming a new connection is a very fast task, like the friends network on facebook. On the other hand, in others networks we clearly do not expect a densification. For example, a scientist only publishes papers every few month, and even less frequently publishes a paper with a new colleague, with whom he never collaborated before. Thus it would be very surprising, if a sparsification of the collaboration network could be shown.

Because we use a comparitvely small sample size in our test, we want to The insertion of a single vertex can be interpreted as a Galton-Watson process: It is kind of a survivor effect This can be used to prove, that every graph with $p < 0.5$ has to be sparse. We are able to derive a comparitively simple CDF for the upper bound of the extinction probability.

However there are still some open topics, which were not answered in this thesis. In principle, the Forest Fire Model could be defined with any heavy-tailed distribution as rondom number generator. Thus one could substitute the geometric distributon by poisson or exponential distribution. The only variant we tested so far was a constant fraction. However, the results are h unuseable results (mainly because they were too random) We have only considered non-directed graphs. What if one considers directed graphs with loops. Is it also possible to derive an upper bound like in our case.

Feel free to visit `https://github.com/AlexanderAlt/Thesis-Complex-Networks/` if you are interested in doing your own tests with the program, or want the cee the data with hich the diagrams we created.

# Bibliography


[1] Aaron Clauset, Cristopher Moore, and M. E. J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453:98–101, 2008. doi: 10.1038/nature06830. URL http://dx.doi.org/10.1038/nature06830.

[2] Paul Erdos and Alfred Renyi. On the evolution of random graphs. *Publ. Math. Inst. Hungary. Acad. Sci.*, 5:17–61, 1960.

[3] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data*, 1(1):2, 2007. ISSN 1556-4681. doi: http://doi.acm.org/10.1145/1217299.1217301. URL http://portal.acm.org/citation.cfm?id=1217299.1217301.

[4] M. Newman. *Networks: An Introduction*. OUP Oxford, 2010. ISBN 9780199206650. URL https://books.google.de/books?id=q7HVtpYVfC0C.

[5] M.E.J. Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003. ISSN 0036-1445. doi: 10.1137/S003614450342480.

[6] Ravi Srinivasan. Topics in complex networks @ONLINE, 2013. URL http://www.ma.utexas.edu/users/rav/ComplexNetworks/.