Explosive Percolation

Universität Leipzig

Cameron Perot

July 2019

Abstract

Contents

Chapter 1

Introduction

1.1 What is Percolation Theory?

Graphs are a fundamental part of nature, therefore the study of them is of great importance in a number of different fields. A graph can be anything from a random network composed of points (nodes) and connections (edges) between them to a lattice or set of points in a periodic arrangement. To study such things we need a framework in which to do so; this is where percolation theory enter the picture. Percolation theory is in the intersection of multiple other topics such as graph theory, probability theory, and statistical physics, combining them in a way to give us the tools we need to study how groups (clusters) of points behave and evolve within a a network or structure. Percolation can be observed in a variety of environments ranging from the spread of a forest fire to the dissemination of information across a communications network. We often use percolation theory in physics to help us model how something dissipates or flows through a system, e.g. the electrical conductivity or porousness of a material. Some of the applications outside of physics include testing the durability of a transportation infrastructure or determining to what extent a product or idea might be adopted by society. Percolation theory is a broad subject with the ability to be applied to problems in many different disciplines [1] [2].

To get a better understanding of what percolation is, let us discuss a few examples. Society is developing faster than ever before due to the propagation of information and goods through networks we have constructed. These networks then allow us to create and participate in increasingly more advanced networks, thus it is critical that we study and understand how these networks behave so that we can get an idea of how to manage and improve them. The most well known being the internet, which sustains a massive flow of information between billions of people across the globe. When something is posted online one might ask: how many people will it reach? To answer this one would start at the source node then see which nodes it could possibly be connected to. If each person has a probability p of sharing it with another person then we can construct a probabilistic model to determine what the cluster could possibly look like. If the probability is high then we would expect the size of cluster to be of significant size when compared to the size of the whole network.

Another interesting application is that to the spread of health epidemics across a population. If there is a virus going around there is an underlying network that can be studied; there is the initial person(s) who contracted the virus and then the people who were contaminated from coming in contact with those infected. Thus the obvious question is: how much of the population will it affect? Let us say in a very basic sense that an infected person has a p percent chance of infecting those they come into contact with. If the virus is treatable and p is low, then we would expect the virus to be eradicated fairly quickly since doctors have the time and ability to treat the outbreak properly. On the other hand if p is high, say higher than some critical probability p_c, then we might expect a different scenario where the virus spreads across the population, i.e. percolates, much like the Ebola epidemic in West Africa during 2013-2016. Therefore, percolation theory plays a crucial role in helping us answer important questions when it comes to determining how something might spread or propagate across a network or structure.

1.1.1 Phase Transitions

To understand the most important use of percolation theory we must first understand what a phase transition is. In physics we often encounter systems which have different properties depending on which phase they're in. Thus we are interested in how the system behaves around the critical point. A phase transition is usually characterized by an order parameter, which is a variable that is zero in one phase and non-zero in the other, giving us the ability to distinguish phases and identify phase transitions.

In percolation theory we define the order parameter as the ratio of the largest cluster to the size of the system, i.e. if we have n nodes total and the largest cluster C contains $|C|$ nodes, then the order parameter P is given by:

$$P := \frac{|C|}{n} \tag{1.1}$$

Of course the system is usually examined in the thermodynamic limit ($n \to \infty$), thus P is zero unless there exists a macroscopic cluster.

There are two main types of phase transitions: first- and second-order. In a first-order phase transition there is what we call a latent heat, i.e. energy that is either absorbed or emitted during a process where the temperature is fixed, e.g. the transition of water between the solid and liquid phases. Second-order (also called continuous) phase transitions are characterized by a divergent correlation length and power law behavior of variables which leads to a set of critical exponents (more about critical exponents below). The reason they are called continuous transitions is because the order change in the order parameter at the critical point is continuous. The classic percolation models are known to undergo continuous phase transitions, making them useful for modeling such systems.

Critical Exponents

Around the critical point in a percolation model, there exists a set of numbers called the critical exponents which contain information about how certain quantities of interest behave. Let's take for example the order parameter as a function of the occupation probability p, which is described by the critical exponent β [1]:

$$P(p) \sim (p - p_c)^{\beta} \tag{1.2}$$

The average cluster size $\langle s \rangle$ is described by γ [1]:

$$\langle s \rangle (p) \sim (p - p_c)^{-\gamma} \tag{1.3}$$

The correlation length ξ is described by ν [1]:

$$\xi(p) \sim |p - p_c|^{-\nu} \tag{1.4}$$

The neat thing about these exponents is that they are universal in the sense that they only depend on the dimension d of the system, regardless of the microscopic configuration. Some values for these are given in the table below [1]:

d	β	γ	ν
2	5/36	43/18	4/3
3	0.41	1.82	0.88

1.1.2 Percolation on a Lattice

On a lattice we can look at percolation from two different perspectives: site and bond percolation. With site percolation we study graphs where the sites of a lattice are either occupied or unoccupied, whereas with bond percolation we study the graphs where the connections between sites are either active or inactive. The underlying concepts of percolation are the same, but in practice the details differ slightly.

Site Percolation

For this we consider a lattice of dimension d. Let us take our lattice to be hyper-cubic with side length L, giving us a total of L^d sites in the lattice. We then occupy each site of the lattice with probability p (unoccupied with probability $1-p$), independent of all other sites. We can then use indicator random variables $X_1, ..., X_{L^d}$ to represent if the sites are occupied or not, where:

$$X_i = \begin{cases} 1 & \text{if site } i \text{ occupied} \\ 0 & \text{otherwise} \end{cases}$$

The number of occupied sites in the lattice is then given by $\sum_i X_i$, which is expected to be $L^d p$. The notion of a cluster $I = \{i_1, ..., i_s\}$ of size $s = \sum_{i \in I} X_i$ is defined as a set of s occupied sites which are nearest neighbors with at least one other site in the set. If we let N_s represent the number of clusters of size s, and $n_s = N_s / L^d$, then we can also look at the probability that any given site is part of a cluster:

$$\sum_s s n_s \tag{1.5}$$

This now gives us the ability to determine the probability that the given site is part of a cluster of size s:

$$\frac{s n_s}{\sum_{s'} s' n'_s} \tag{1.6}$$

Using the above we can now compute the average cluster size:

$$\langle s \rangle = \sum_s s \frac{sn_s}{\sum_{s'} s'n'_s} \tag{1.7}$$

Thus it is clear that we can obtain a significant amount of information by applying basic probability theory. Now that we have defined clusters and cluster sizes we can talk about the idea of a percolating cluster. This is a cluster which spans from one side of the lattice to the opposite in any of the d dimensions. It's obvious that the presence of a percolating cluster is heavily dependent on p, since for low p we wouldn't expect to see a cluster spanning across the lattice. However, this is also dependent on L, and we often seek to study the limiting case where $L \to \infty$. This leads us to the concept of a critical probability, p_c, such that when $p > p_c$ we expect to see a percolating cluster, but not when $p < p_c$.

One-Dimension We now consider the simplest case where $d = 1$, which allows us to analytically solve for some quantities of interest. Fig. 1.1 illustrates a snippet of an infinite 1D lattice where the solid black sites are active and the white ones are inactive. There are a total of four clusters visible (highlighted in blue): two one-clusters, a three-cluster, and a four-cluster.

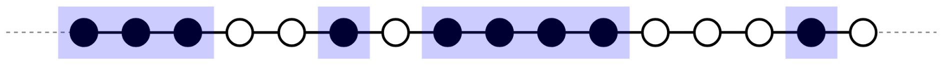

Figure 1.1: 1D Site Clusters

For comparison Fig. 1.2 illustrates a 1D lattice where all sites are active and only some of the bonds between them are active. There are a total of four clusters visible (highlighted in blue): two two-clusters, a three-cluster, and a four-cluster.

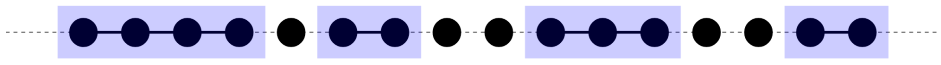

Figure 1.2: 1D Bond Clusters

The probability that any given site is part of a cluster of size s is then given by the product of the independent probabilities of the site to the left of the leftmost site in the cluster being unoccupied, s consecutive sites being occupied, and the site to the right of the rightmost site in the cluster being unoccupied, i.e.:

$$n_s = (1 - p) \cdot p^s \cdot (1 - p) = p^s (1 - p)^2 \tag{1.8}$$

Plugging Eq. 1.8 into Eq. 1.5 we find:

$$\begin{aligned}\sum_s n_s s &= (1-p)^2 \sum_s s p^s \\ &= (1-p)^2 p \frac{d}{dp} \sum_s p^s \\ &= (1-p)^2 p \frac{d}{dp} \frac{p}{1-p} \\ &= (1-p)^2 p \left[\frac{1}{1-p} + \frac{p}{(1-p)^2}\right] \\ &= p\end{aligned} \tag{1.9}$$

Where the second equality is obtained by linearity of the derivative and summation operators and realizing that $sp^s = p\frac{d}{dp}p^s$, the third by noticing that $\sum_s p^s$ is the power series for $\frac{p}{1-p}$, the fourth by application of the derivative operator, and the final by algebaic manipulation. Therefore there is a p percent chance that any given site is part of a cluster, which when thought about is fairly obvious. Using the above we can also analytically determine the average cluster size (again using the trick of differentiating p^s to bring down an s):

$$\begin{aligned}\langle s\rangle &= \sum_s \frac{s^2 n_s}{\sum_{s'} s' n'_s} \\ &= \frac{(1-p)^2}{p} \sum_s s^2 p^s \\ &= \frac{(1-p)^2}{p} \left[p\frac{d}{dp}\right]^2 \sum_s p^s \\ &= \frac{(1-p)^2}{p} \left[p\frac{d}{dp}\right]^2 \frac{p}{1-p} \\ &= \frac{(1-p)^2}{p} p\frac{d}{dp} \left[\frac{p}{1-p} + \frac{p^2}{(1-p)^2}\right] \\ &= \frac{1+p}{1-p}\end{aligned} \tag{1.10}$$

The above illustrates that we can analytically solve for some of the above quantities of interest in the one-dimensional case. We're often more interested in the thermodynamic limit, that is for systems that is, when $L \to \infty$. In the thermodynamic limit, percolation on the chain only occurs when we have an infinite number of neighboring occupied sites. This means that if just one site out of the entire chain is unoccupied there is no percolating cluster, which leads us to conclude that for the one-dimensional case $p_c = 1$.

Multiple-Dimensions Unfortunately it is not as straight forward in higher dimensions due to the many different arrangements and shapes that the clusters can take on. This is illustrated in Fig. 1.3 where we can see four possible arrangements of a nine-cluster, each with different numbers of neighboring inactive sites.

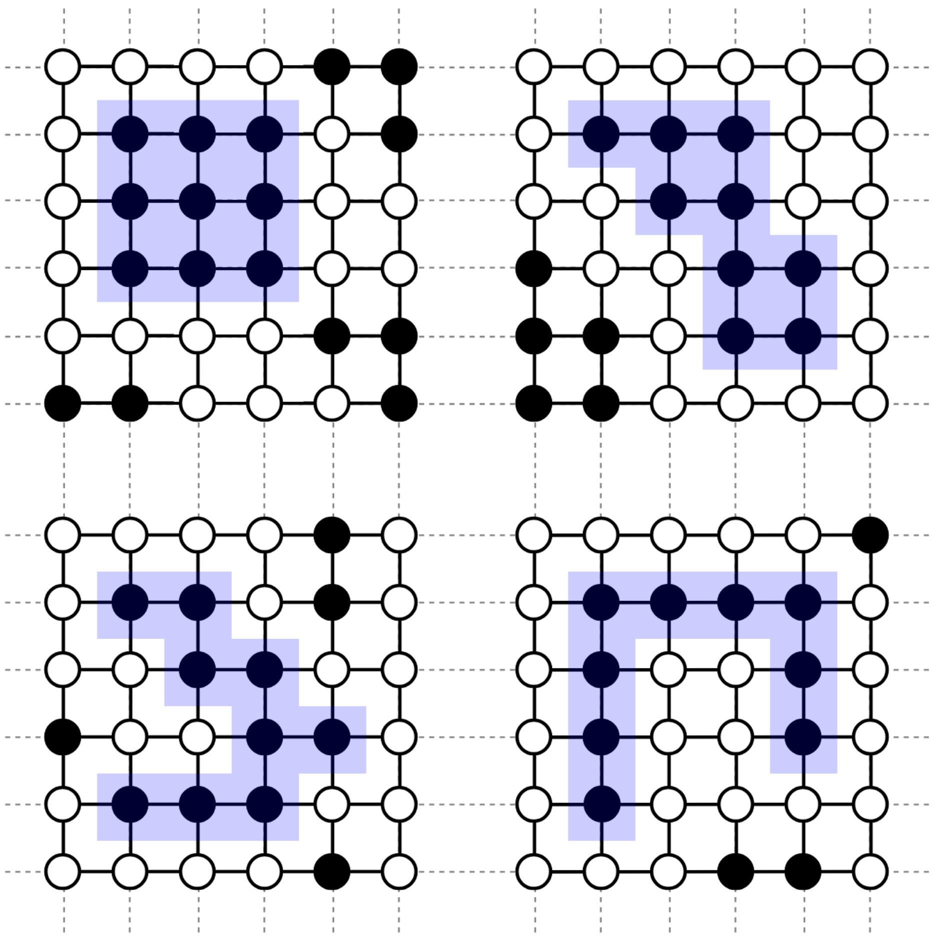
Figure 1.3: 2D Site Clusters

Therefore, we lose our ability to answer our questions through analytical methods and we must resort to numerical methods and simulations to study the system. This is where computer simulations and finite-size scaling analysis come in handy. Due to the limitations of computational power we can only simulate systems up to certain sizes, however, we would still like to know gather some information about the infinite size system. What we do is we simulate systems of varying sizes and see how the quantities of interest scale with the system size, which gives us the ability to extrapolate to the thermodynamic limit.

Example: Ising Model The Ising model is the most well-known model of ferromagnetism, undergoing a second-order phase transition (for $d \geq 2$) from the ordered, ferromagnetic state at low temperatures to the disordered, paramagnetic state at high temperatures. This model has played a critical role in the development and understanding of statistical physics. The basic idea is that given a lattice of size L^d, at each site i of the lattice there is a spin σ_i oriented in either the up or down direction. The order parameter characterizing the transition is the magnetization $m = \frac{1}{L^d}\sum_i \sigma_i$. A (simplified) Hamiltonian of the system in the configuration $\{\sigma\}$ is then given by:

$$H(\{\sigma\}) = -J\sum_{\langle ij\rangle}\sigma_{ij} - h\sum_i \sigma_i \tag{1.11}$$

Where the sum is over all neighboring spin pairs, J is the coupling constant between spins, and h is a mean field per spin approximation. Using this combined with statistical and computational physics methods we are able to simulate how this system behaves as the size increases. One useful tool when modeling such a system is the use of periodic boundary conditions, which help reduce the finite-size effects of smaller lattice sizes. In Fig. 1.4 we can see a percolating cluster of up (black) spins at the critical temperature $T_c = 2/\log(1+\sqrt{2})$ on a 2D square lattice with side-length $L = 128$. Fig. 1.4 was generated using the Wolff single cluster algorithm with $5 \cdot 10^4$ update sweeps over the lattice.

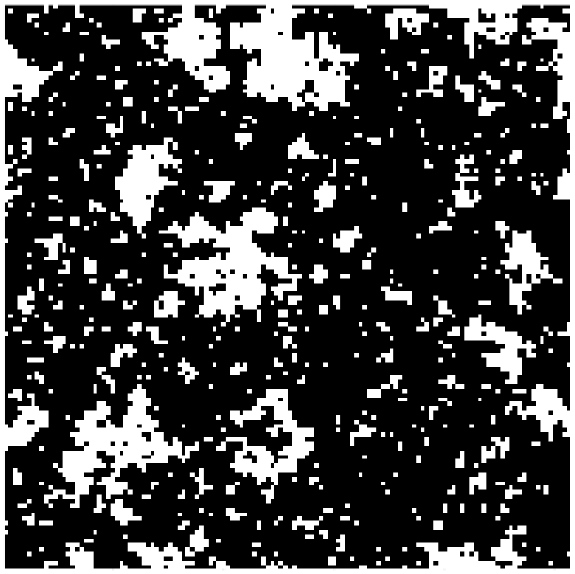
Figure 1.4: 2D Ising Model Percolating Cluster

1.1.3 Percolation on a Network

Percolation can also be observed on networks, so we will now discuss some of the basic ideas behind network percolation as a segue to the next section. In the late 1950s and early 1960s Paul Erdős and Alfréd Rényi published several papers on random networks, leading to the creation of what is known today as the Erdős-Rényi model (ER) [3] [4].

To get an idea of how it works we imagine a graph starting with a set of n disconnected nodes at $t = 0$, giving us $\binom{n}{2}$ possible connections between them. Then at each step in the evolution process two nodes are chosen at random and connected by an edge, thus after t steps there are a total t edges present in the graph.

A cluster on a random network is a set of nodes connected by edges, either directly or indirectly. Being that some clusters in the graph will of course merge together over time, there exists an attractive potential of sorts between clusters corresponding to the likelihood that the clusters will merge together to form a larger one. To demonstrate this let's take three clusters C_1, C_2, and C_3 of sizes $|C_1|, |C_2|$, and $|C_3|$, with $|C_1| > |C_2|$ leaving $|C_3|$ arbitrary. Then we let Φ_1 (Φ_2) be the potential between the first (second) cluster and the third. In a graph where each connection is equally probable, the probability that the first or second cluster merges with the third is directly proportional to the size of the cluster because more nodes means more possible connections. Therefore, there is a higher probability that the first cluster rather than the second merges with the third and we conclude that $\Phi_1 > \Phi_2$.

In the a network, percolation occurs when a giant (macroscopically large) cluster appears. The order parameter here is defined as the largest cluster size divided by the network size, i.e. $|C|/n$. Let n be the number of nodes, t be the number of edges present in the graph, and take $r = t/n$. If $r > 0.5$ then there will exist a percolating cluster within the graph, but not if $r < 0.5$ [4], thus $r_c = 0.5$ is the critical point where the system transitions from the disconnected state to the state of large scale connectivity. Letting $|C|$ be the size of the largest cluster, it can be shown that for $r < 0.5$ the largest cluster size scales as $C \sim \log n$, and for $r > 0.5$ $C \sim n$, and more specifically if $r \gtrsim 0.5$ one finds $C \approx (4r - 2)n$ [4]. Therefore, the ER model exhibits a continuous phase transition when the number of edges exceeds half the number of nodes in the graph. Looking at Fig. 1.5 we can see that the order parameter is zero up until $r = 0.5$ when it transitions to the state of large scale connectivity.

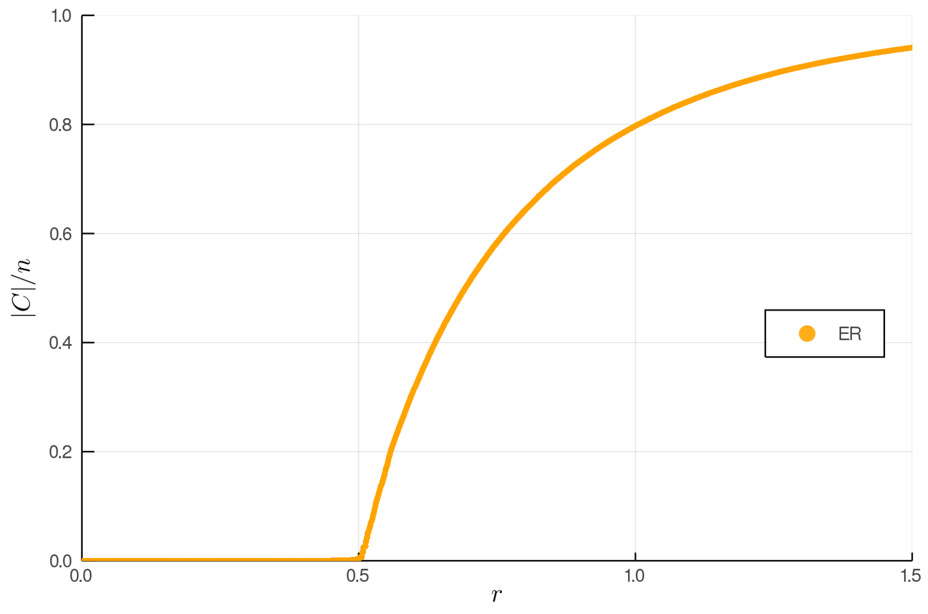

Figure 1.5: Erdős-Rényi Model Order Parameter, $n = 10^6$

1.2 An "Explosive" Transition

In this paper we are going to take a deeper look into a specific type of percolation called explosive percolation, where the underlying concept is that the onset of percolation is delayed until a certain point where it then occurs at an accelerated rate. This occurs when the underlying evolution process works in such a way that the largest cluster size $|C|$ is controlled. We can think of a graph where multiple clusters might evolve separately without merging. Collectively they take up a large portion of the graph but there isn't a percolating cluster yet due to the lack of connections between the them. If at some point these clusters do begin to connect then the graph transitions to the percolating state. For certain evolution processes it has been heavily debated whether or not the phase transition is continuous or not, so the aim of the next section is to summarize what we know to this point regarding the nature of the transition for said processes.

1.2.1 A Selective Process

This all started in the year 2000 when Dimitris Achlioptas raised an interesting question about how a graph of n nodes would evolve under certain conditions [5]. First we suppose that at each step t in the evolution process two edges e_t and e'_t are evaluated, but only one of them is selected to be added to the graph. Which of the proposed edges is selected is based only on the information contained in $e_1, e'_1, ...e_{t-1}, e'_{t-1}$. Does there exist an algorithm for adding edges to the graph such that with high probability a giant component does not appear until $t/n > 0.5$? This process of evaluating $m \geq 2$ edges at each step is now referred to as an m-edge Achlioptas process. In the next few sections we will see how a small change to the way edges are added can have a big impact on the evolution of the system.

1.2.2 The First Look

In 2001 Tom Bohman and Alan Frieze were the first to design and analyze an Achlioptas process in their paper "Avoiding a Giant Component" [5], where they set out to answer Achlioptas' question. This paper laid out the framework for the Bohman-Frieze (BF) model of graph evolution and showed that there does exist a process in which the appearance of a giant cluster is delayed; so the answer to Achlioptas' question is yes! This can be seen in Fig. 1.6 where the order parameter for the BF model remains smaller than for the ER model, but a little after $r = 0.5$ it begins to rise at a faster rate than in the ER model, eventually overtaking the ER model.

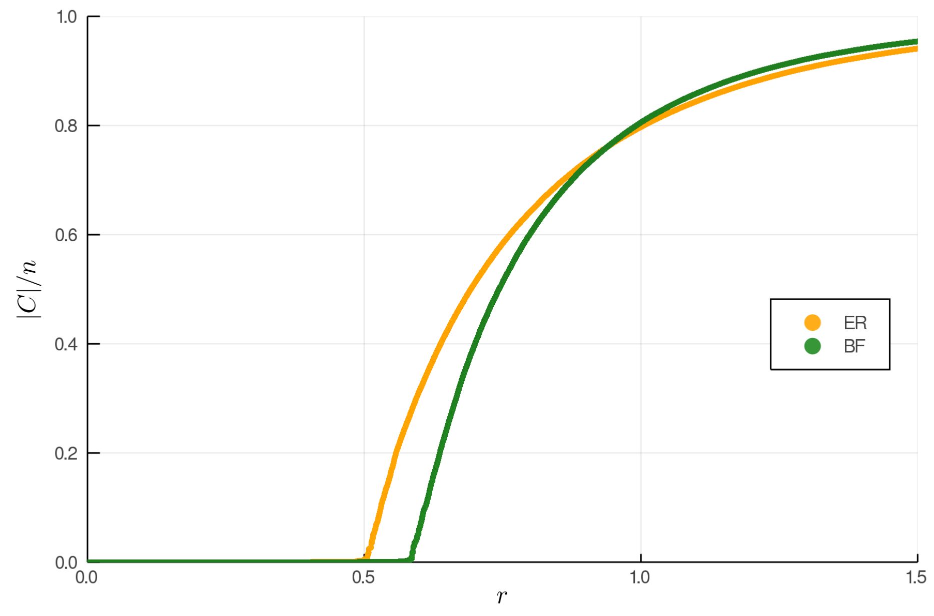

Figure 1.6: Bohman-Frieze Model Order Parameter, $n = 10^6$

The algorithm they came up with randomly/uniformly samples and evaluates two edges e_t and e'_t at each step t, adding e_t if it connects two isolated nodes and discarding e'_t, otherwise adding e'_t and discarding e_t. This is known as a bounded-size rule, which has since been generalized to a K bounded-size rule where e_t is added if it connects clusters of size smaller than K, otherwise e'_t is added. What this process does is equally discriminate against clusters of size greater than or equal to K. It has been hypothesized that all bounded-size rules exhibit continuous phase transitions [6].

A form of the algorithm can be written in pseudocode as follows:

- Let T be the total number of edges to add to the graph.
- Let $A_t = \{e_1, e_2, ..., e_t\}$ be the set of accepted edges at step t.
- Let e_t^1 and e_t^2 be the two nodes which edge e_t would connect.
- Let $C(e_t^i)$ be the cluster which e_t^i belongs to.

Algorithm 1 Bohman-Frieze

```
procedure BF($T, K = 2$)
    $A \leftarrow \emptyset$
    $t \leftarrow 1$
    while $t \leq T$ do
        if $|C(e_t^1)| < K$ and $|C(e_t^2)| < K$ then
            $A \leftarrow A \cup \{e_t\}$
        else
            $A \leftarrow A \cup \{e'_t\}$
        end if
        $t \leftarrow t + 1$
    end while
end procedure
```

A few years after their original paper, Bohman and Frieze published another paper with Nicholas Wormald titled "Avoidance of a giant component in half the edge set of a random graph" [7]. In this paper they designed a new algorithm (BFW) for adding edges to a graph that runs in phases beginning at $k = 2$. Starting with n isolated nodes, edges are randomly/uniformly sampled and evaluated one at a time. If the edge being evaluated would create a cluster of size less than or equal to k then the edge is accepted, otherwise the edge is rejected. If edge is rejected and the ratio of accepted edges is less than the function $g(k) = 1/2 + \sqrt{1/(2k)}$, then the algorithm moves to the next phase.

A form of the algorithm can be written in pseudocode as follows:

- Let T be the total number of edges to add to the graph.
- Let $A_i = \{e_1, e_2, ..., e_i\}$ be the set of accepted edges at step i.
- Let $C(A)$ be the largest cluster in A.

Algorithm 2 Bohman-Frieze-Wormald

```
procedure BFW($T$)
    $A \leftarrow \emptyset$
    $k \leftarrow 2$
    $t \leftarrow 1$
    $i \leftarrow 1$
    while $t \leq T$ do
        if $|C(A \cup \{e_i\})| \leq k$ then
            $A \leftarrow A \cup \{e_i\}$
            $t \leftarrow t + 1$
            $i \leftarrow i + 1$
        else if $|A|/i < g(k)$ then
            $k \leftarrow k + 1$
        else
            $i \leftarrow i + 1$
        end if
    end while
end procedure
```

1.2.3 A Discontinuous Transition?

The first mention of explosive percolation appeared in 2009 in the paper "Explosive Percolation in Random Networks" [8] by Dimitris Achlioptas, Raissa M. D'Souza, and Joel Spencer, which will henceforth be referred to as Achlioptas et al. In this paper they laid out a method of choosing edges called the product rule (PR). At each step t in the evolution process two edges are selected randomly/uniformly and evaluated based on the product of the cluster sizes that the edges would connect. More specifically, if edge e_t connects clusters $C(e_t^1)$ and $C(e_t^2)$ and edge e'_t connects clusters $C(e_t'^1)$ and $C(e_t'^2)$, then e_t is accepted if $|C(e_t^1)| \cdot |C(e_t^2)| < |C(e_t'^1)| \cdot |C(e_t'^2)|$, otherwise e'_t is accepted.

A form of the algorithm can be written in pseudocode as follows:

- Let T be the total number of edges to add to the graph.
- Let $A_t = \{e_1, e_2, ..., e_t\}$ be the set of accepted edges at step t.
- Let e_t^1 and e_t^2 be the two nodes which edge e_t would connect.
- Let $C(e_t^i)$ be the cluster which e_t^i belongs to.

This is illustrated in Fig. 1.7 where the first proposed edge e_t would connect the blue cluster of size 5 to the red cluster of size 3 and the second proposed edge e'_t would connect the orange cluster of size 2 to the green cluster of size 4. The product corresponding to e_t is $5 \cdot 3 = 15$ whereas the product corresponding to e_t is $2 \cdot 4 = 8$, so e_t is rejected and e'_t accepted. If we were looking at Fig. 1.7 from within the framework of the BF model with $K = 1$, then e_t would be rejected and e_t accepted even though both the orange and green clusters are of size greater than K.

Algorithm 3 Product Rule

```
1:  procedure PR($T$)
2:      $A \leftarrow \emptyset$
3:      $t \leftarrow 1$
4:      while $t \leq T$ do
5:          if $|C(e_t^1)| \cdot |C(e_t^2)| < |C(e_t'^1)| \cdot |C(e_t'^2)|$ then
6:              $A \leftarrow A \cup \{e_t\}$
7:              $t \leftarrow t + 1$
8:          else
9:              $A \leftarrow A \cup \{e'_t\}$
10:             $t \leftarrow t + 1$
11:         end if
12:     end while
13: end procedure
```

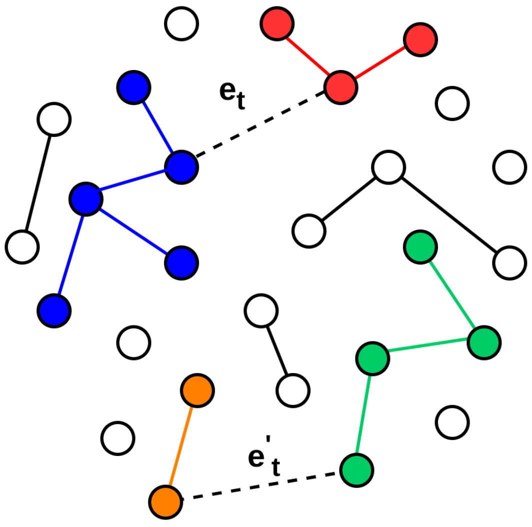

Figure 1.7: Edge Selection

Fig. 1.8 shows the order parameters for the ER, BF, and PR models, making it clear just how drastically different the PR model transition is than in the ER/BF models. As can be seen the order parameter remains near zero for much longer until the critical point $r_c \approx 0.888...$ [8], which is much later than the critical point in the ER model of $r_c = 0.5$. Not long after the critical point is reached the order parameter appears to jump up discontinuously, which was a surprising find.

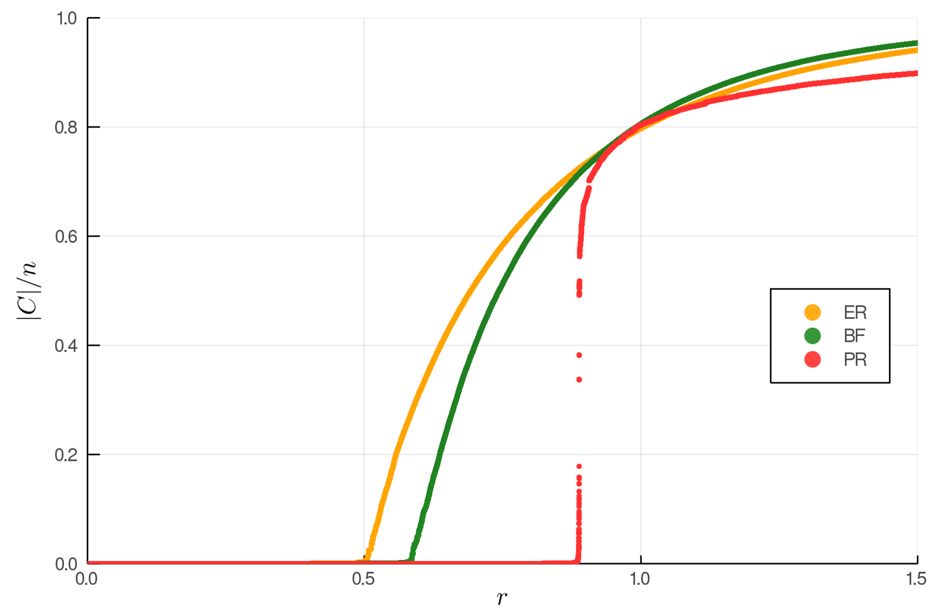

Figure 1.8: Product Rule Model Order Parameter, $n = 10^6$

In their paper they analyzed the scaling behavior of their model, defining the quantity $\Delta := t_1 - t_0$, where t_0 is the last step where $C < n^{1/2}$ and t_1 is the first step where $C > 0.5n$. Δ is linear in n for continuous transitions (i.e. an extensive quantity), however, their analysis indicated otherwise that $\Delta < 2n^{2/3}$ and $\Delta/n^{2/3} \to 1$ in their simulations for sizes up to $6.4 \cdot 10^7$. To arrive at these results they took an ensemble average of 50 different independent and identically distributed observations and fit a curve to determine the relation between Δ and n. Their findings are illustrated in Fig. 1.9. [8]

Fig. 2. (**A**) The ratio Δ/n for ER and BF for increasing system sizes. (**B**) The ratio $\Delta/n^{2/3}$ for PR for increasing system sizes. (**C**) Convergence to r_c = 0.888... from above and below (the two curves fitted independently). (**D**) A linear scaling relation is obeyed in the range $\gamma \in [0.2,0.6]$, shown here for A = 0.5. Color shows convergence with increasing system size n to the relation $\gamma + 1.2\beta = 1.3$. Our numerical experiments establish this scaling relation for $A \in [0.1,0.6]$ and we expect that in larger system sizes this range of A would broaden, particularly the lower end.

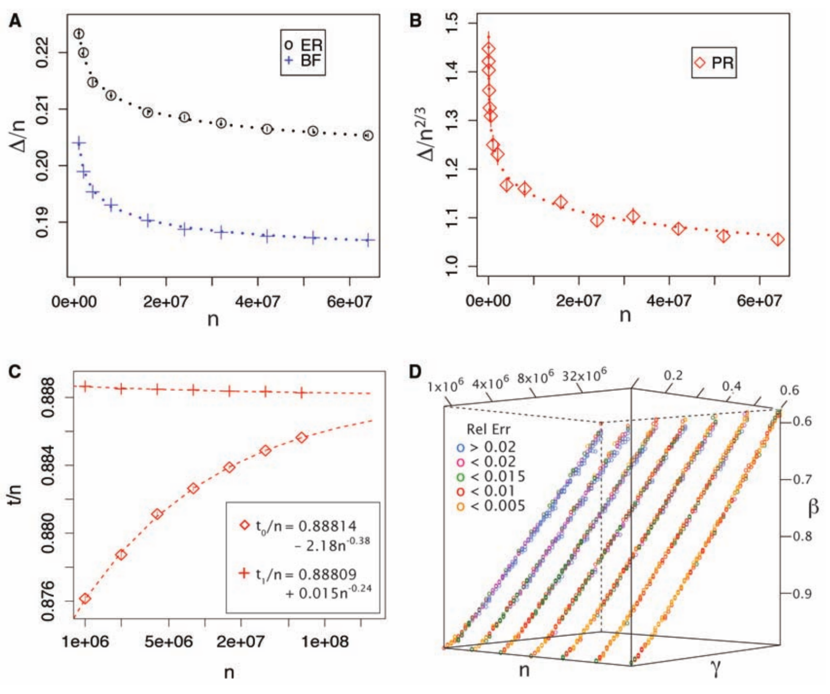

Figure 1.9: Achlioptas et al. Simulation Results, Source: [8]

They ended their paper by saying: "We have demonstrated that small changes in edge formation have the ability to fundamentally alter the nature of percolation transitions. Our findings call for the comprehensive study of this phenomenon, and of its potential use in bringing phase transitions under control." [8]

No, It's Actually Continuous!

20100915

Well, Maybe Not...

20110315

Continuity With Unusual Finite-Size Scaling Behavior

20110322

Large Scale Simulations

20110616

The Power of Mathematical Statistics

20120821

Supercritical BFW Phase Transitions

20130524

Critical Exponents in Explosive Percolation

20140505

"Solution of the explosive percolation quest"

20150212 20150331

A Look At the Developments

20150701

Two Types of Discontinuous Percolation

20150710

Chapter 2

A New Method

Chapter 3

Visualization and Analysis

Chapter 4

Conclusion

Appendix A

Appendix

A.1 Designing the Library

I chose Julia as the language to write the library in for several reasons, the largest being that I believe Julia is the perfect language for running large scale distributed computational physics/mathematics simulations. The benefit of being able to use it as an interpreted language while at the same time getting speeds of a compiled language can not be understated since the immediate feedback and ability keep a kernel running with data in RAM allows for easy debugging and data exploration. Julia code revolves around its type system which works nicely with its use of multiple dispatch, and at the heart of the type system is defining custom types and creating hierarchies with which functions can take specific types. The type hierarchy of Julia's default types is illustrated in Fig. A.1.

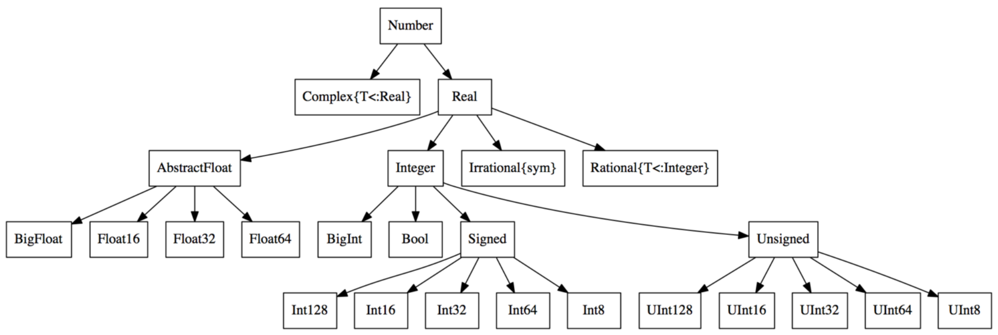

Figure A.1: Julia Type Hierarchy

The type system is actually opt-in, meaning that one need not type their variables, but typing can help with performance as well as gaining a better understanding of how the code works.

I started off with creating custom types to represent networks/lattices, the type hierarchy of them is illustrated below in Fig. A.2.

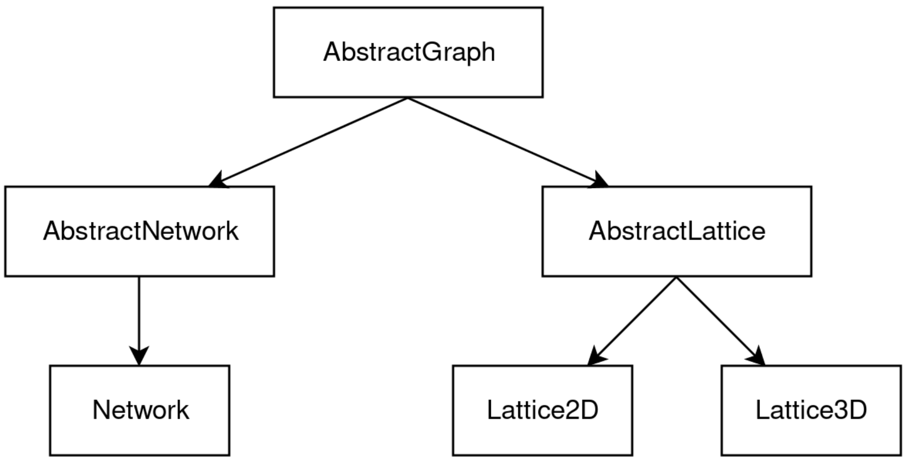

Figure A.2: Custom Type Hierarchy

As of now there are three different types of graphs: `Network`, `Lattice2D`, and `Lattice3D`. The `Network` type represents an undirected network where edges can be present between any two unique nodes. The `Lattice2D` type represents a two-dimensional square lattice where edges can be present only between nearest neighbors, similarly where `Lattice3D` represents a three-dimensional cubic lattice. The following is a brief conceptual overview of how the system functions, for more detailed descriptions and explanations of methods or variables please refer to the documentation.

Each of these types have a similar structure in the way they store information about the underlying graph they represent. To instantiate a type one must pass it the size and optionally a seed value for the random number generator (which is used in the evolution for randomly selecting edges), which in a `Network` means passing the number of nodes `n` and in a `LatticedD` passing the side length `L` ($\mathtt{n} = \mathtt{L}^{\mathtt{d}}$). The types also have a variable called `t` which refers to the step in the evolution process and represents the number of edges present in the graph. The edges present in the graph are kept track in the `edges` variable, which is a set of two-tuples of integers representing an edge between node `i` and `j` as `(i, j)`. As of now the edges in all types are undirected i.e. `(i, j) = (j, i)`, but only `(i, j)` is stored to minimize unnecessary RAM usage.

Keeping track of the clusters as they evolve is the meat of these simulations, therefore it is necessary to speak about how this is done. `cluster_ids` is a `d`-dimensional array `(d = 1 (Network), 2 (Lattice2D), 3 (Lattice3D))` which contains the information relating to which cluster a node is in. `clusters` is a dictionary where the keys are the cluster IDs and the values are sets of nodes which are in the given cluster. `cluster_sizes` is a dictionary where the the keys are the cluster IDs and the values are integers representing the number of nodes in the given cluster. The associated observables are stored in a custom type called `Observables` under the name `observables`.

Now that we understand how the clusters are tracked let's talk about how this is done in reality. Let `t` be the current step in the evolution process where edge `(i, j)` has been chosen to be added to the graph. Let $C_{\mathtt{i}}$ represent the cluster to which node `i` is a member of, i.e. $\mathtt{i} \in C_{\mathtt{i}}$. We then look and determine which cluster is smaller and merge the smaller cluster into the larger cluster in-place (this small trick saves a significant amount of computational time), e.g. if $|C_{\mathtt{j}}| < |C_{\mathtt{i}}|$ then all of the nodes in $C_{\mathtt{j}}$ will have their cluster ID updated to that of $C_{\mathtt{i}}$ and the set of nodes representing $C_{\mathtt{i}}$ will be updated to include the nodes in $C_{\mathtt{j}}$, and the entry for $C_{\mathtt{j}}$ will then be deleted from `clusters` to reduce the memory footprint.

After the clusters are merged the associated observables are updated, which currently includes the largest cluster size, the average cluster size, the cluster heterogeneity (number of unique cluster sizes), and some post-simulation analysis quantities for identifying where the phase transitions and if it is continuous.

A.1.1 Making Use of Multiple Dispatch

Multiple dispatch allows us to define multiple functions with the same name, yet different code depending on the input type. This is extremely useful for dealing with differences between networks and lattices such as when selecting an edge, because for example in a `Network` we can select any inactive edge between nodes, whereas in a `LatticedD` we can only select an inactive edge between nearest neighbors (this can be seen implemented in `edge_methods.jl`).

Another benefit of this coupled with the type system is that when we have a function that is generic enough to act on both `Network`s and `LatticedD`s then we can set the input type for that function to `AbstractGraph` or if it's general enough to act on all `LatticedD`s but not `Network`s then we can set the input type to `AbstractLattice`. This allows for consistency, e.g. we can call `prod-uct_rule!(Network(Int(1e6)))` and similarly `product_rule!(Lattice2D(Int(1e3))` and the code will decide based on the type which functions to use.

A.1.2 Running a Simulation

Running a simulation is simple, one must first download the `Percolation.jl` library, which can be found on GitHub under the MIT license. Then, to run a simulation on a `Network` with 10^6 nodes using the product rule to add $1.5 \cdot 10^6$ edges, and view the largest cluster size time series data we would run:

```
# add the library to the load path and import it
push!(LOAD_PATH, "/path/to/Percolation.jl/src")
using Percolation

# instantiate the Network
g = Network(Int(1e6))

# run the evolution process
product_rule!(g, Int(1.5e6))

# show the largest cluster size time series data
g.observables.largest_cluster_size
```

Bibliography

[1] D. Stauffer and A. Aharony. *Introduction to Percolation Theory.* 2nd ed. Philadelphia, Pennsylvania: Taylor and Francis, 2003.

[2] M. Sahimi. *Applications of Percolation Theory.* 1st ed. Philadelphia, Pennsylvania: Taylor and Francis, 1994.

[3] P. Erdős and A. Rényi. "On random graphs, I." In: *Publ. Math. Debrecen* 6 (1959), pp. 290–297.

[4] P. Erdős and A. Rényi. "On the evolution of random graphs". In: *Publ. Math. Inst. Hungar. Acad. Sci.* 5 (1960), p. 17.

[5] T. Bohman and A. Frieze. "Avoiding a giant component". In: *Random Structures Algorithms* 19 (2001), pp. 75–85.

[6] J. Spencer and N. Wormald. "Birth control for giants". In: *Combinatorica* 27(5) (2007), pp. 587–628.

[7] T. Bohman, A. Frieze, and N. C. Wormald. "Avoidance of a giant component in half the edge set of a random graph". In: *Random Struct. Algorithms* 25(4) (2004), pp. 432–449.

[8] D. Achlioptas, R. M. D'Souza, and J. Spencer. "Explosive Percolation in random networks". In: *Science* 323(5920) (2009), pp. 1453–1455.