Modeling explosive percolation on directed networks

Shannon Moran

PHYS 514, Fall 2017

1 Motivation

For my final project, I was particularly motivated by taking what we'd learned and applying it to a problem recently discussed in the literature. Part of the value I see in graduate courses is that they provide you with the vocabulary to go out and understand related literature without studying the exact problems in class.

The paper I chose to investigate was "Explosive percolation on directed networks due to monotonic flow of activity", published in PRE by the D'Souza and Lu groups in July 2017 [1]. In this paper, the authors looked to replicate the observed growth of tweets in response to the discovery of the Higgs boson (this data set is publicly available as part of the Stanford Large Network Dataset Collection, hence the interest). These authors saw the growth of the tweet network and recognized it as **explosive percolation**.

Explosive percolation is said to occur in an evolving network when a macroscopic connected component emerges in a number of steps that is much smaller than the system size, and there have been a number of papers (e.g. [2]) that have explored this phenomena in random networks.

I particularly liked this paper because it took topics that we had covered in class and extended them, as shown in Figure . I will detail the implementation of these processes in the Methods section, and discuss a few specific findings in the Results and Discussion section. Note that this brief report is also accompanied by a jupyter notebook of code examples, annotated code used in this project, and the presentation given on December 13.

2 Methods

2.1 Percolation processes

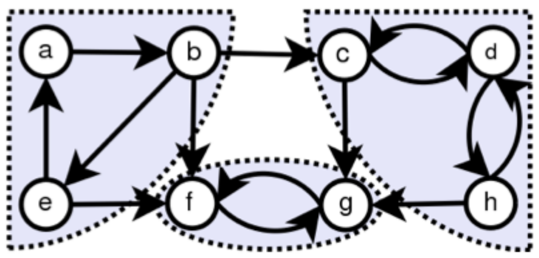
The percolation processes used in the paper are based on the Directed Erdos-Renyi (DER) proces, which is a stochastic, unbiased network generation algorithm. Here, the authors add in an order (ODER) and competition plus order (C-ODER) rule. These algorithms are summarized in Figure 2.

Each network is characterize by n nodes and m edges. We can also combine these into a measure called "edge density", $\delta = \frac{m}{n}$. Each directed edge points from a *tail* to a *head*.

2.2 Cluster definition and algorithms

In lattice percolation, we're familiar with clusters defined as occupied sites connected to occupied nearest neighbor sites. In random network percolation, we refer to networks of nodes connected by edges as components. In percolation on directed networks, we study strongly connected components (SCCs). Nodes a and b are considered in the same SCC if a can reach b and b can reach a via directed edges.

This is illustrated below:[1]



Algorithms for finding clusters tend to be highly problem-specific. While we used the Hoshen-Kopelman algorithm for lattices, and Newman-Ziff is used for undirected graphs, we use Tarjan's Algorithm for finding the SCCs of directed graphs [3]. This is a depth-first search. The order of operation is:

- Two parameters are tracked for each node, v
  - v.index = time stamp at which vertex v is discovered
  - v.low = oldest ancestor v can reach
- For each node:
  - Add the node to the stack
  - For each child:
    - If child has already been visited: recursively run the SCC-finding function, an set v.low = min of the node's low and the child's low
    - Elif child is in the current stack: v.low is the minimum of node's low and child's low
    - If v.low and v.index are equal, the stack down to the node is one SCC

[1]Source: https://commons.wikimedia.org/wiki/File:Scc.png

Because of this recursion, I run into stack overflow issues and am unable to study networks with greater than `n=1E3` nodes. Additional information, as well as an implementation using an open-source SCC-finding algorithm, is provided in the jupyter notebook.

2.3 Critical exponents

With their competitive, ordered process, the authors are interested in comparing their results with prior work on competitive percolation networks. In a 2011 *Nature Physics* paper, Nagler et al demonstrated that the largest jump in the size of the largest component of a system would scale as $N^{-\beta}$ [4].

However, they find that "the only exception seems to be global competition, where [they] find β to be indistinguishable from zero". That is precisely what Waagen et al find– or claim to find– for both the ODEr and C-ODER processes.

To study this, we must find the single link addition that generates the largest jump in the size of the largest SCC. Recalculating after the addition of every edge would be prohibitively computationally expensive. Instead, they build out the graph, tracking the order in which edges were added, then use a binary search on the completed graph to find the largest jump. With this process, the cost of the operation is only $O(E \log E)$.

3 Results and Discussion

3.1 Studying the growth of the largest SCC: Paper figures 5, 6, and 8

As shown in Figure 3, I am able to replicate the shapes of both the ODER and C-ODER curves of largest SCC size versus edge density, including a series of large jumps near the "critical edge density" (δ_c) for the C-ODER study. However, my graphs are significantly shifted towards smaller edge densities, though when I increase the number of nodes by a factor of 10, they begin to shift to the right.

This immediately tells me that the δ_c is not system-size invariant. While the authors frame edge density as an analog to the probability p of a site being occupied in lattice percolation, this is misleading. In lattice percolation, p_c of there being a percolating cluster in the system is invariant to system size– it is just the shape of the distribution that changes with system size (wider distribution for smaller systems, a step function for an infinite system).

Not to leave this stone unturned, I then looked for a variable that would capture the probability of a large cluster existing relative to the number of edges added, without varying with number of nodes. My most promising idea was to study the fraction of a system connected to an average node ($\frac{\delta}{n}$) rather than edge density. However, as is obvious from Figure 4, this was a bust.

Finally, it is worth noting that this shift is not due to a failure in my implementation's ability to replicate the mechanisms of SCC formation. As shown in Figure 5, I am able to replicate the mechanism of the largest SCC "jump" at the δ_c by the combination of the two largest SCCs.

3.2 Studying the critical exponent: Paper figure 7

As shown in Figure 6, I am able to replicate the authors' finding that the critical exponent of the growth of the size of the largest cluster with system size appears to be zero. Even at my much smaller system sizes, I get a similar average size of "largest jump" relative to system size.

4 Conclusions

Not only did I learn about network theory in doing this project– a field I hadn't studied before– I also was able to use my understanding of the percolation methods we discussed in class to understand and then implement analogous methods in networks (rather than lattices).

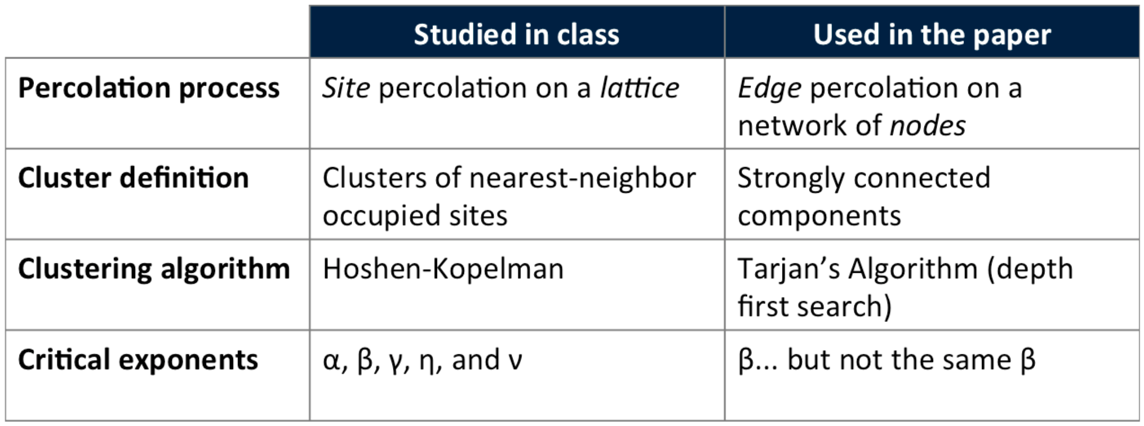
References

[1] Alex Waagen, Raissa M. D'Souza, and Tsai-Ching Lu. Explosive percolation on directed networks due to monotonic flow of activity. *Physical Review E*, 96(1), Jul 2017.

[2] Dimitris Achlioptas, Raissa M. D'Souza, and Joel Spencer. Explosive percolation in random networks. *Science*, 323(5920):1453–1455, 2009.

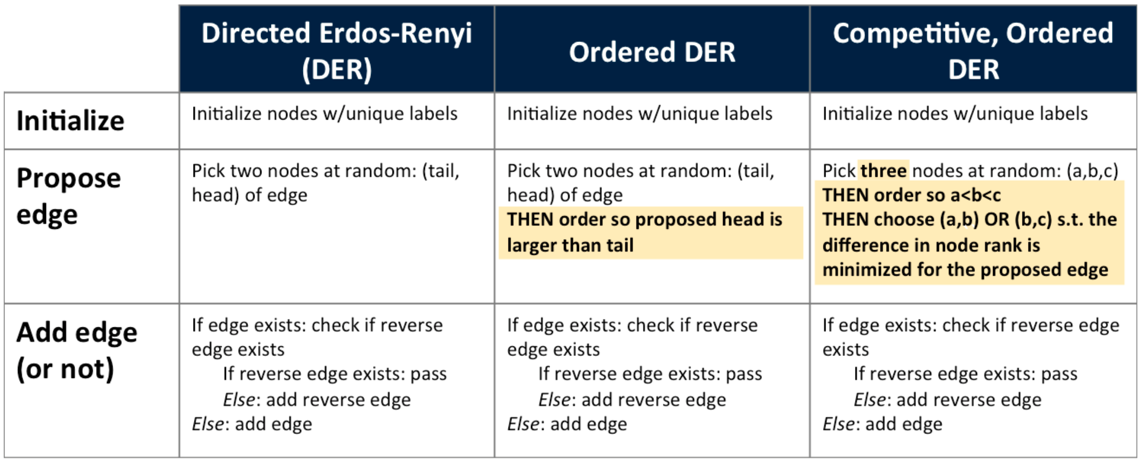
[3] Robert T. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2), 1972.

[4] Jan Nagler, Anna Levina, and March Timme. Impact of single links in competitive percolation. *Nature Physics*, 7:265–270, 2011.



	Studied in class	**Used in the paper**
Percolation process	*Site* percolation on a *lattice*	*Edge* percolation on a network of *nodes*
Cluster definition	Clusters of nearest-neighbor occupied sites	Strongly connected components
Clustering algorithm	Hoshen-Kopelman	Tarjan's Algorithm (depth first search)
Critical exponents	α, β, γ, η, and ν	β... but not the same β

Figure 1: Comparison of topics studied in class, versus those discussed in the paper by Waagen et al.



	Directed Erdos-Renyi (DER)	Ordered DER	Competitive, Ordered DER
Initialize	Initialize nodes w/unique labels	Initialize nodes w/unique labels	Initialize nodes w/unique labels
Propose edge	Pick two nodes at random: (tail, head) of edge	Pick two nodes at random: (tail, head) of edge **THEN order so proposed head is larger than tail**	Pick **three** nodes at random: (a,b,c) **THEN order so a<b<c** **THEN choose (a,b) OR (b,c) s.t. the difference in node rank is minimized for the proposed edge**
Add edge (or not)	If edge exists: check if reverse edge exists If reverse edge exists: pass *Else*: add reverse edge *Else*: add edge	If edge exists: check if reverse edge exists If reverse edge exists: pass *Else*: add reverse edge *Else*: add edge	If edge exists: check if reverse edge exists If reverse edge exists: pass *Else*: add reverse edge *Else*: add edge

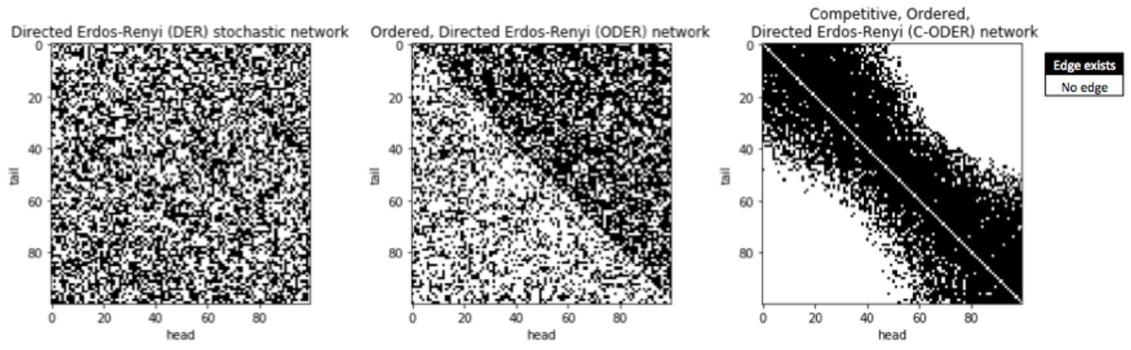


Figure 2: Explanation of percolation processes covered in Waagen et al., and the adjacency matrices resulting from their implementation.

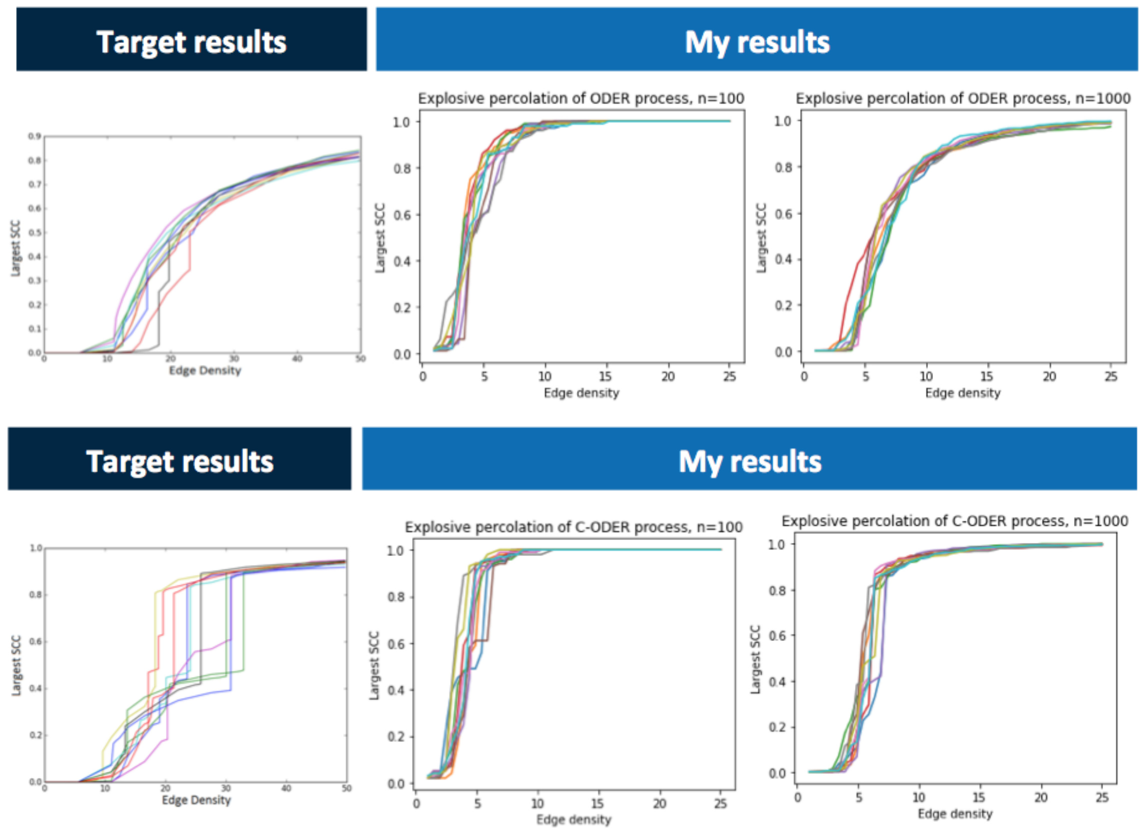


Figure 3: Replication of Figures 5 (top, for the ODER process) and 6 (bottom, for the C-ODER process) from Waagen et al. Both the paper and my simulations were run for 10 replicates, though Waagen et al use a system size of n=1E6. My system sizes are as indicated.

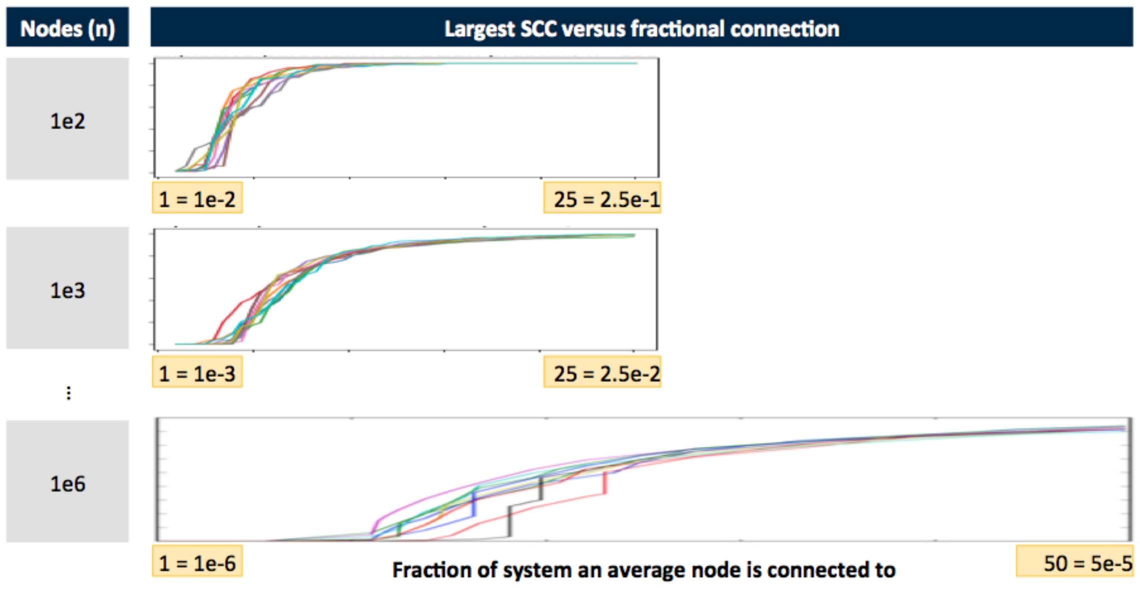


Figure 4: Illustration of using fraction of network connected to an average node, rather than edge density, as the x-coordinate.

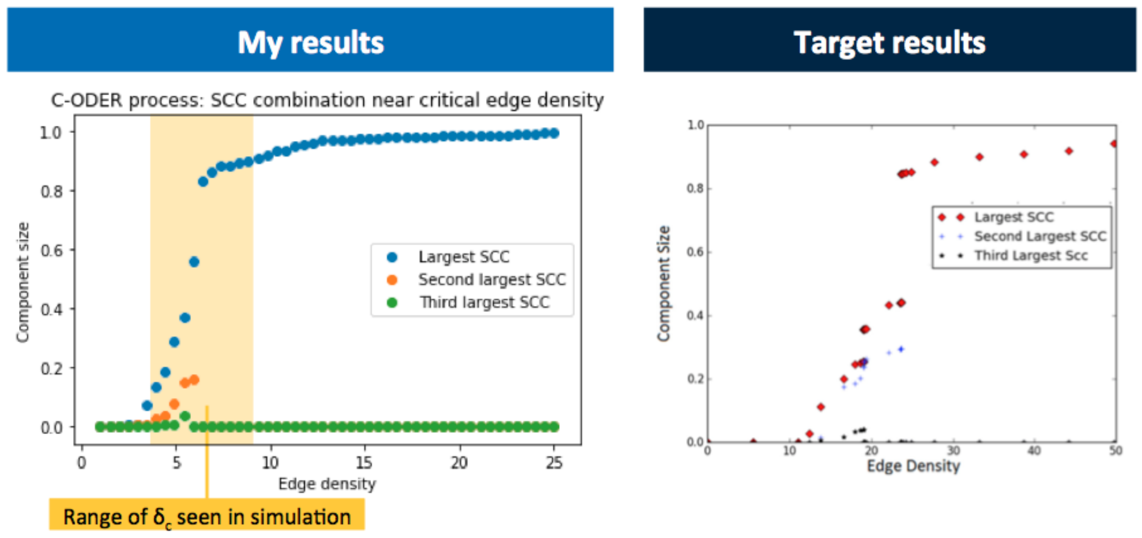


Figure 5: Replication of Figure 8 from Waagen et al. While the paper uses 1 replicate at a system size of n=1E6, I use 1 replicate at a system size of n=1E2.

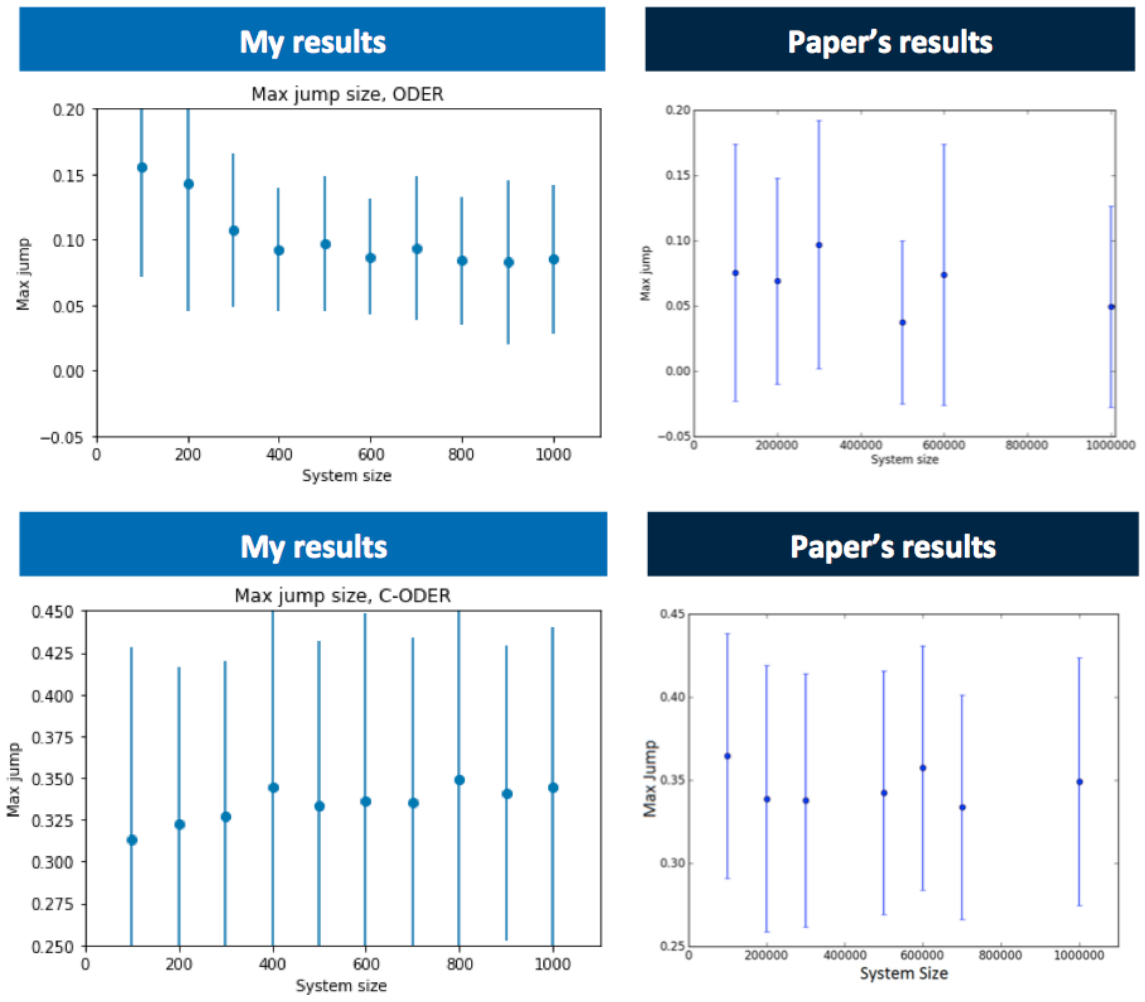


Figure 6: Replication of Figure 7 from Waagen et al. Both the paper and my results represent the average and standard deviation of 40 replicates. I assumed an edge density of 50; the edge density used in the paper figures was not stated.