Opinion Formation on Clustered Networks and Its Implications

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
BACHELOR OF ARTS
WITH HIGH HONORS IN
MATHEMATICS

by

Amanda M. Fritz

Advisor: Nishant Malik
Department of Mathematics
Dartmouth College, Hanover, NH USA

2016

Contents

Abstract

The dynamics of the spread of contagions on networks are often studied employing models which are not sensitive to the structural properties of the underlying networks. I propose a new model that allows us to study the influences of network structure on the dynamics occurring on the network. I employ the voter model on a Watts-Strogatz network of 500 or 1000 nodes and examine how networks with different levels of clustering in them influence the processes involved in collective opinion formation. I then apply my new model to analyze structural properties of real-world networks, in particular Facebook. The goal is to develop dynamics on networks as a tool for analyzing the structure of real-world network data sets.

Acknowledgements

I would like to thank my advisor, Dr. Nishant Malik, for helping me work through and complete my research. I would not know about networks or nearly as much about coding without his assistance and encouragement. I would also like to thank the Dartmouth College Department of Mathematics for supporting me and giving me funding for my poster. Thank you to my friends for their support throughout the last few months. Finally, I thank my parents, my brother, and my sister for their encouragement and for listening to me talk about this project for the past two terms. I could not have done this without you!

Chapter 1

Introduction

“Mathematicians do not study objects, but the relations between objects.”

- Henri Poincaré

Networks exist everywhere. They are part of everyday life, whether realized or not. There are biological networks, such as the metabolic system, technological networks, such as the World Wide Web or power grids, and social networks. The most commonly conceived social networks are social media networks, such as Facebook, Twitter, Instagram, or Pinterest, however there are also less well-known social networks, such as friend groups and sexual contact networks [17]. The study of networks, otherwise known as network theory, is a topic of great research and interest among quantitative scientists as they study various aspects of networks, such as their structural properties, their generative processes, and various kinds of dynamics on them. Next I will discuss a few basics about networks.

A network consists of nodes that are attached by edges. The edges can be directed, undirected, or weighted. Networks can have non-trivial local and global clustering, both originating from the tendency of nodes to form triangles of connections— in other words, the tendency of nodes to link with neighbors of a neighbor [17]. This is especially true for social networks, as it mimics the phenomenon of the high probability of two friends also having a common friend. Furthering the notion of clustering are communities, in which groups of nodes are more highly connected to those within their group than to those in outside groups [12].

Networks show a wide range of complex topological properties, ranging from having completely random connectivity to completely regular connectivity [17]. There has been much interest in studying a variety of dynamics on complex networks, especially the spread of different kinds of contagions on such networks [1–10, 12–16, 18, 20–26]. In this work I will be studying the spread of social contagions, namely opinions, and my main emphasis will be on studying the role of network structure on the process of collective opinion formation. In this thesis I will limit myself to the study of the influence of clustering on opinion formation, in particular global clustering or transitivity and not local clustering or average local clustering.

Such studies are important in the context of the rising role of social media in forming opinions. It is of great importance to identify aspects of social networks that may inhibit or encourage the progress of certain views or opinions [20, 26]. Opinion formation is a complex process involving many behavioral aspects; here I neglect many of them and try to build a minimalistic model which only includes some of the essential aspects of opinion formation. Though my model is minimalistic in nature, I observe that it is able to reproduce many of the complex features observed in opinion formation.

To generate different network topologies I use the Watts-Strogatz model. In this network model, nodes connect to neighboring nodes with no double edges or self edges, and there exists a rewiring probability, which also determines the net clustering in the network [17]. The Watts-

Strogatz model has two main parameters: the average degree $\langle k \rangle$ and the rewiring probability p, where $p \in \{0,1\}$ with $p = 0$ forming a ring, i.e., every node has the same number of nearest neighbors, and with $p = 1$ forming a completely random graph [7, 17]. The formula for the clustering coefficient for the Watts-Strogatz model is $C = \frac{3\times(\langle k \rangle - 1)}{2\times(2\langle k \rangle - 1)} \times (1-p)^3$ [24]. This explains why there is connected ring topology when $p = 0$, since the clustering coefficient C is at a maximum. As p varies, one crosses a scenario in which average path length is small and clustering is high— this case is also known as *small world* [22].

I use the well-known voter model, founded independently by Clifford and Sudbury and Holley and Liggett, for simulating opinion formation [5]. The voter model has been a very popular choice for simulating opinion dynamics, as it can generate a wide variety of dynamical features and real-world scenarios [4, 6, 10, 13]. For example, Holme and Newman uncovered a non-equilibrium transition point between two contrasting consensus states in a model with multiple opinions and adaptive topology [10, 13]. Durrett et. al. identified critical parameters involved in the transition between consensus states with different rewiring strategies [4].

In my model, I do not include rewiring, i.e., adaptive evolution of the network. Rather, voter dynamics are made to evolve on a subgraph of the given initial graph. This choice provides me with a couple of advantages. The first is that the influence of initial network structure can be studied in detail. The second is that this model can be directly run over any given network data sets. The second advantage provides an unique opportunity to employ this model in order to study the structures of real-world networks. There have not been many attempts to study network data using these dynamic models.

In addition, my model is more pragmatic in nature, as we do not have examples of real-world data sets with adaptive network evolution. Apart from that, most models with an adaptive network component only have a random rewiring step, which means that all the results obtained on such models are only applicable to a specific kind of random network, namely Erdös-Rényi random networks.

The influence of basic network properties on contagion dynamics has served as the motivation for empirical studies. One such significant study was performed by Damon Centola [2]. His experiment resembled online dating. He observed social behavior and how it affects opinion formation [2]. In it he investigated the hypothesis about opinion formation that contagions spread faster on random networks than on clustered ones [2]. He found, contrary to the usual intuition, that contagions spread faster on networks with clustering [2]. His main reason to explain this observation was that the existence of strong social reinforcement in networks with clustering helps in spreading the contagions faster [2].

Social reinforcement is not a structural property of networks, but a behavioral one. Here in this work I try to understand whether or not only the structural properties of a clustered network can make contagions spread faster on such networks. For this purpose, I propose to employ the Watts-Strogatz network to study the voter dynamics. I simplify the network to have binary opinions 0 or 1, remove any social influence factors, and execute my model on a subgraph of a given network. I can control clustering C by varying the rewiring probability p in the Watts-Srogatz network model. Therefore, by using this network model, I can study the influence of clustering on the opinion dynamics. An additional goal of this work is to use the voter dynamics to analyze the structural properties of real-world networks. I employ my new model on Facebook data, obtained from the Stanford Large Network Dataset Collection (SNAP) [11].

Chapter 2

The model

"The sciences do not try to explain, they hardly even try to interpret, they mainly make models. By a model is meant a mathematical construct which, with the addition of certain verbal interpretations, describes observed phenomena. The justification of such a mathematical construct is solely and precisely that it is expected to work. "

- John von Neumann

2.1 Algorithm

Let $G_0(N,L)$ be a network with N nodes and L edges. In my model G_0 is a Watts-Strogatz network, but when applied to Facebook data G_0 is the Facebook network. Let O be an opinion vector of length N, where $O_i \in \{0,1\}$. N_1 represents the density of total nodes holding opinion 1, whereas N_0 represents the density of total nodes holding opinion 0. $\langle k \rangle$ stands for average degree. p is the rewiring probability, and it determines the net clustering in the network— I examine all $p \in \{0,1\}$ in increments of 0.1.

Algorithm 1 Voter dynamics on subgraphs of static networks

Generate a network $G_0(N,L)$ of given topology with N nodes and L links.
Assign each node an opinion, i.e. $O = [O_i]_{i=1}^{i=N}$, where $O_i \in \{0,1\}$.
for $\alpha \in \{0.1, 0.9\}$ **do**
  Randomly select a subgraph $G_1(N, \lambda L) \subset G_0$.
  λL edges in G_1 are referred to as **active**.
  Calculate L_{01} for G_1, referred to as L_{01}.
  while $L_{01} \neq 0$ **do**
    Randomly choose a $L_{ij} \in L_{01}$.
    Generate a uniform random number $x \in \{0,1\}$.
    if $x > \alpha$ **then**
      $O_i \to O_j$.
    else
      Find the subgraph $G_{01}(N,(1-\lambda)L) \subset G_0$ with edges $(1-\lambda)L$ not in G_1 and in G_0, referred to as **inactive**.
      Calculate L_{01} for G_{01}, referred to as L'_{01}.
      Randomly choose a $L_{ab} \in L'_{01}$.
      Remove edge L_{ij} from G_1, making it inactive, and insert L_{ab} into G_1, making it active.
    Calculate $\rho = \frac{\sum n}{N}$, where, for $n \in N$, $n = N_0$ if $\sum N_0 < \sum N_1$ and $n = N_1$ if $\sum N_1 < \sum N_0$.

In my model I first select a subgraph $G_1 \subset G_0$, where G_1 has the same number of nodes as G_0 but fewer edges. Every edge that is included in $G_1 \subset G_0$ is called **active**, whereas remaining edges in G_0 are referred as **inactive**.

In my model a node accepts an opinion of its neighbor with probability $1-\alpha$, and an edge turns from active to inactive with probability α, i.e., it is removed from $G_1 \subset G_0$. To keep the number of edges conserved in $G_1 \subset G_0$ at all times, I also turn one of the inactive edges into active at random, i.e., add one edge into $G_1 \subset G_0$. I call α the *transformation probability*, as the transformation of an edge from active to inactive depends on α.

An edge is called discordant if it connects two nodes with opposite opinions. The density of discordant edges is represented by L_{01}. Similarly, harmonious edges connect nodes with the same opinion. Their density is represented by L_{00} and L_{11}. ρ is defined as the fraction of nodes with the minority opinion. In other words, if 0 is the majority opinion then ρ gives the fraction of nodes holding opinion 1.

2.2 Further details

I first create a Watts-Strogatz network, G_0, with N nodes, either $N = 500$ or $N = 1000$, and L total edges. I evenly distribute randomly assigned opinions, 0 or 1, amongst the nodes. I make a fraction of edges from G_0, λL, as **active** and turn those active edges into a subgraph, $G_1 \subset G_0$. The size of $G_1 \subset G_0$ depends on into how many subgraphs G_0 is divided— let λ represent this division. The fraction of edges $(1-\lambda)L$ that are in G_0 but not in $G_1 \subset G_0$ are **inactive**— the subgraph created from these edges is called $G_{01} \subset G_0$. We define the the set of discordant edges in $G_1 \subset G_0$ to be $\{L_{01}\}_{i=1}^{i=\lambda L}$ and the set of discordant edges in $G_{01} \subset G_0$ to be $\{L'_{01}\}_{i=1}^{i=(1-\lambda)L}$.

Within these active and inactive edges exist both harmonious edges, L_{00} and L_{11}, and discordant edges, L_{01}, that together sum to L. The relation of these edges can be written as: $L = L_{00} + L_{11} + L_{01}$, where L_{01} accounts for both discordant edges 0 — 1 and 1 — 0. I then randomly pick up a discordant edge, $\{L_{01}\}_{i=h} \in G_1 \subset G_0$, and with probability $1-\alpha$ flip the opinion $O_i \to O_j$. I randomly choose a discordant edge, $\{L'_{01}\}_{i=h} \in G_{01} \subset G_0$ and make that edge active while simultaneously making $\{L_{01}\}_{i=h}$ inactive. Thus, now $\{L_{01}\}_{i=h} \notin G_1 \subset G_0$ and $\{L'_{01}\}_{i=h} \in G_1 \subset G_0$. I continue this process until $L_{01} = 0\ \ \forall \alpha \in \{0.1, 0.9\}$ in increments of 0.1 and $\forall p \in \{0.0, 1.0\}$ in increments of 0.1, which corresponds to changing values of C. Note that I do not include $\alpha = 0$ or $\alpha = 1$. This is because at $\alpha = 0$ we flip the opinion $O_i \to O_j$ with probability 1 to make 1 — 0 to 1 — 1 or 0 — 1 to 0 — 0. If we allow for $\alpha = 0$, it is the same as performing the procedure on the entire supergraph G_0, and in this way the opinions will end up being homogenous across the network, thus making it an uninteresting and unrealistic case.

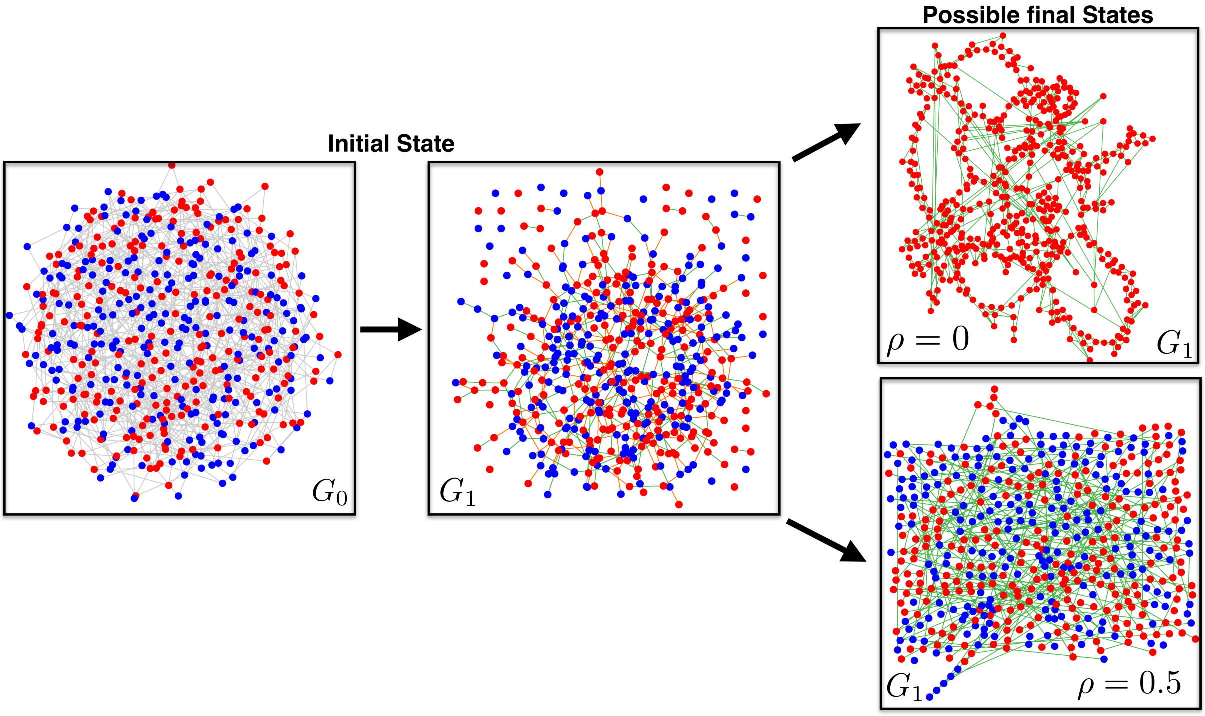

Figure 2.1: Selection of subgraph G_1 from G_0. The $\rho = 0$ case represents unanimous consensus and the $\rho = 0.5$ case represents fractured consensus. Red nodes represent nodes with opinion 1, blue nodes represent nodes with opinion 0, orange edges represent discordant edges, and green edges represent harmonious edges. For the $\rho = 0$ case, the parameters used are: $p = 0$, $\langle k \rangle = 6$, $\lambda = \frac{1}{2}$, and $\alpha = 0.4$. For the $\rho = 0.5$ case, the parameters used are: $p = 1$, $\langle k \rangle = 4$, $\lambda = \frac{1}{3}$, and $\alpha = 0.8$.

There must exist some chance that new active links will be inserted into $G_1 \subset G_0$ in order to make the simulation realistic. At $\alpha = 1$ we only make $\{L_{01}\}_{i=h}$ inactive and $\{L'_{01}\}_{i=h}$ active, thus never reducing the number of L_{01} and never reaching $L_{01} = 0$.

Figure 2.1 illustrates the division of G_0 into nodes of opinion 1 and 0, as well as into active and inactive edges. It shows the subgraph $G_1 \subset G_0$, containing only active edges, and how that subgraph is further divided into nodes of differing opinions based on the resulting ρ value, $\rho = 0$ or $\rho = 0.5$. Red nodes represents nodes with opinion 1 and blue nodes represent nodes with opinion 0. Orange edges represent discordant edges and green edges represent harmonious edges.

One question that arises during simulations is, “What is the cutoff time to determine whether or not the network converges to $L_{01} = 0$?" In order to answer this question, I test different combinations of $\langle k \rangle$ and λ, timing how long each combination of p and α takes to reach $L_{01} = 0$. I define convergence for my model as the following:

$$\lim_{t \to t_F} L_{01}(t) = 0 \text{ , thus converging} \tag{2.1a}$$

$$\lim_{t \to t_F} L_{01}(t) \neq 0 \text{ , thus not converging, or having a } \textit{jammed state} \tag{2.1b}$$

Where at any time t we have the following conservation law for edges:

$$L_{01}(t) + L_{00}(t) + L_{11}(t) = L \tag{2.1c}$$

Note that equation (2.1a) can be described as *converging in finite time*, whereas equation (2.1b) can be described as *non-converging in finite time*, or *converging in infinite time* and can thus be rewritten as:

$$\lim_{t \to \infty} L_{01}(t) = 0 \tag{2.1d}$$

My model only works within *finite time*. The value I set for t_F is the longest time t I observe it takes for any one combination of p and α to reach $L_{01} = 0$. Define $t_F \sim \mathscr{O}(N^2)$. Thus $|t_F| \leq MN^2$, where $M \in \mathbb{R}^+$. In my simulations I test different values for t_F and find that the results remain constant regardless of which value I choose. I allow the model to run up to $t_F = 200,000$ and $t_F = 1,500,000$, in which cases, when $N = 1000$, $M = 0.2$ and $M = 1.5$, respectively.

Chapter 3

Simulation results on a Watts-Strogatz network

"In mathematics the art of proposing a question must be held of higher value than solving it."

- Georg Cantor

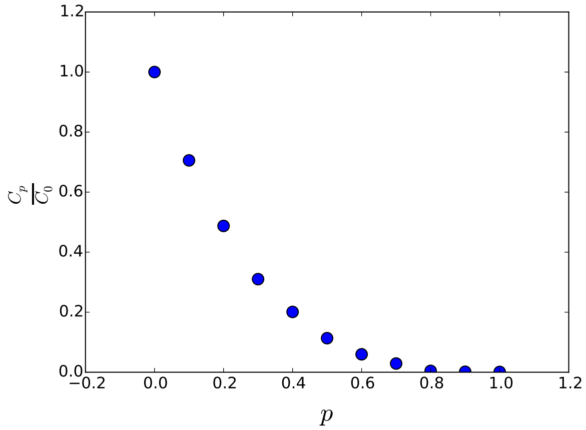

Figure 3.1: ρ versus clustering C in a Watts-Strogatz network of $N = 5000$, $\langle k \rangle = 5$. This graph resembles the characteristic curve for C in a Watts-Strogatz network. C_0 represents the value for C when $p = 0$. C_p represents C for when $p \neq 0$.

In this chapter I will discuss the simulation results. Figure 3.1 shows the characteristic curve for C for the Watts-Strogatz model. Observe as the rewiring probability p increases the clustering C decreases. There exist upper and lower bounds for C:

Upper Bound:

$$C_{upper} = \frac{3 \times (\langle k \rangle - 1)}{2 \times (2\langle k \rangle - 1)} \times (1-p)^3$$

$$C_{upper} = \frac{3 \times (5-1)}{2 \times (2 \times 5 - 1)} \times (1-0)^3$$

$$C_{upper} = \frac{3 \times 4}{2 \times 9} \times 1$$

$$C_{upper} = \frac{12}{18} = \frac{2}{3} = 0.667$$

Lower Bound:

$$C_{lower} = \frac{3 \times (\langle k \rangle - 1)}{2 \times (2\langle k \rangle - 1)} \times (1-p)^3$$

$$C_{lower} = \frac{3 \times (5-1)}{2 \times (2 \times 5 - 1)} \times (1-1)^3$$

$$C_{lower} = \frac{3 \times 4}{2 \times 9} \times 0$$

$$C_{lower} = 0$$

Within these two bounds for C, there exists a critical value for the clustering coefficient C, called C_C, at which the values for ρ transition between 0 and 0.5. I assert that:

$$\exists\, C_C \text{ such that } C < C_C,\ \rho = 0.5 \qquad (3.1a)$$

and

$$\exists\, C_C \text{ such that } C > C_C,\ \rho = 0 \tag{3.1b}$$

I identify C_C to be $C_C \approx 0.25$, since the exact value of C_C is not identifiable unless the simulation is continuously performed over the full range of $p \in \{0,1\}$. One can see this existence of a C_C by examining a graph of α vs. ρ, as shown in Figure 3.2. If ρ depends on α, the dots for each C, as represented by varying colors, would follow the same trajectory, peaking at the same α. What happens in this new model, however, is that each trajectory differs, following very different patterns for different combinations of α and ρ. ρ, in turn, varies with clustering C.

I begin my simulations using $t_F = 200{,}000$. Figure 3.2 is divided into two subplots: (A), when $p \neq 0$, and (B), when $p = 0$. The reason for this is to distinguish between the usual result and the exception. When $p = 0$, C is at a maximum. This means that G_0, and thus $G_1 \subset G_0$, is a perfectly connected graph. Due to this, the simulation happens so fast on such a clustered network that the voter model dynamics never come into play. The results show a highly clustered network with high ρ for each α. By separating out this one case, it is easier to examine how, for higher values of C, you have lower values of ρ.

Another case is that of the *jammed state*, as shown in Figure 3.3. A jammed state is when L_{01} does not converge to 0 before $t = t_F$, or $\lim_{t \to t_F} L_{01}(t) \neq 0$. This graph is interesting because it is observable how many cases reach the limit $t = t_F$ when $t_F = 200{,}000$. For $C < 0.25$, the simulation results in a jammed state. It is notable that for $0.25 < C < 0.45$ we have $0 < \rho < 0.3$, and for $C \geq 0.45$ we have $0.4 < \rho \leq 0.5$. In addition, higher values of C have shorter simulation times t for those cases that do not result in a jammed state. This is what Centola found in

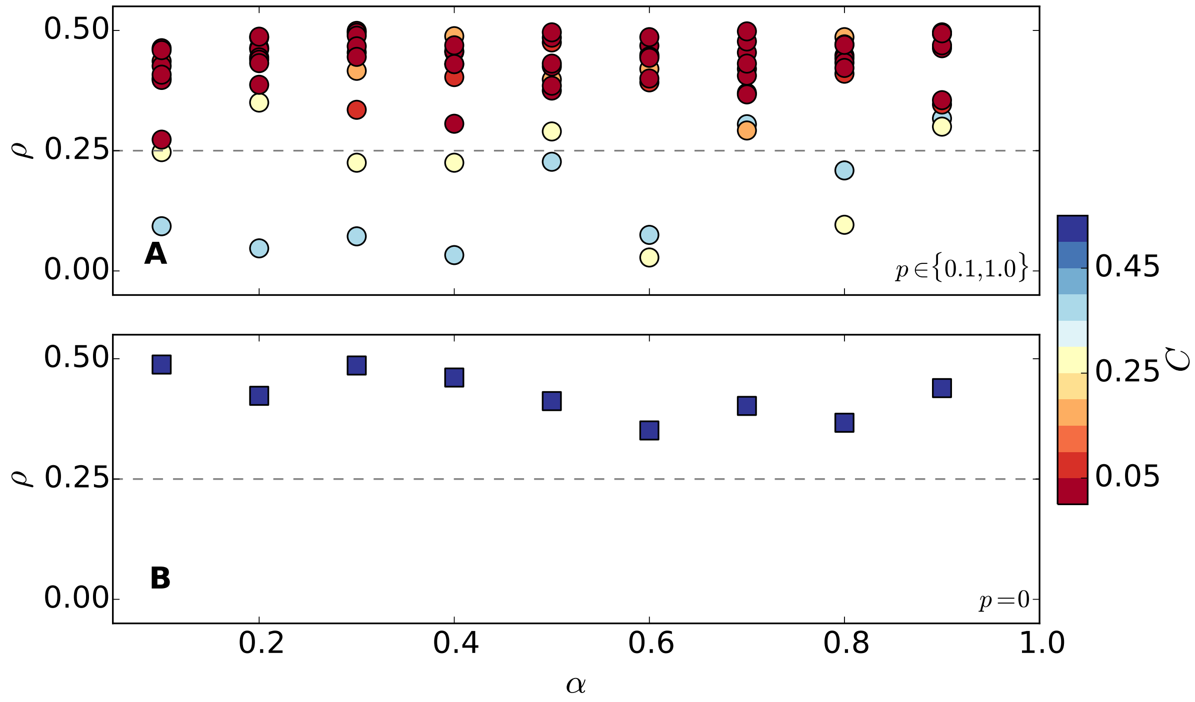

Figure 3.2: Progression of α versus ρ for $N = 1000$, $\langle k \rangle = 5$, $\lambda = \frac{1}{2}$, and $t_F = 200,000$. The colors of the dots represent C. As C decreases, ρ increases except for when $p = 0$ due to the highly clustered nature of the network at that p value.

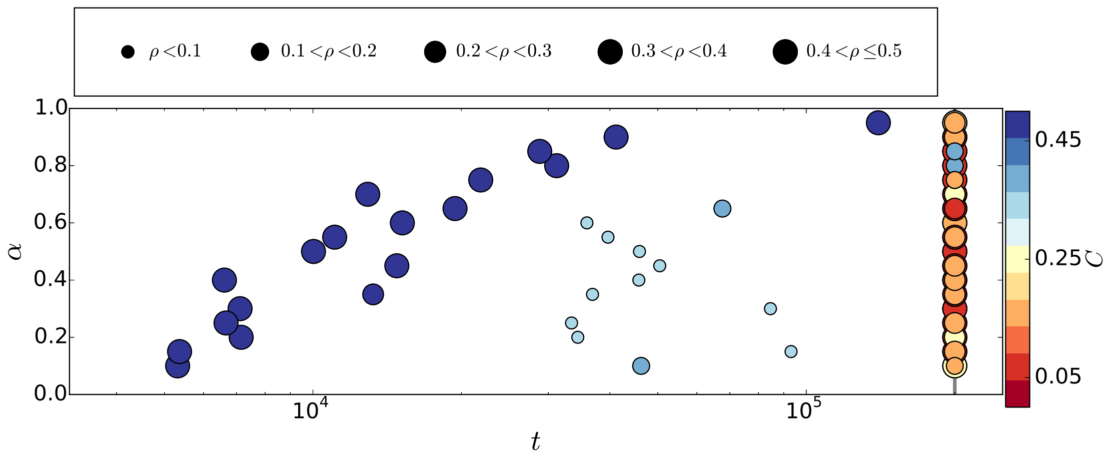

Figure 3.3: Depiction of a jammed state. Progression of t versus α when $N = 1000$, $\langle k \rangle = 5$, $\lambda = \frac{1}{2}$, and $t_F = 200,000$. The color of the dots represent C, while the size of the dots represents ρ. Note the logarithmic scale on t and the ticks for t are in powers of 10. The grey line represents the cutoff time, t_F. When $C < 0.25$, the simulation results in a jammed state. For $0.25 < C < 0.45$ we have $0 < \rho < 0.3$, and for $C \geq 0.45$ we have $0.4 < \rho \leq 0.5$. In addition, the highest C yields the shortest simulation time t.

his empirical study, which resembled online dating, though he accounted this phenomenon to social reinforcement. I have been able to create a similar kind of feature in opinion formation without employing any kind of social reinforcement. Rather I show that one does not need social reinforcement for faster diffusion of opinions on networks with clustering.

One interesting result is that as $p > 0.5$, approximately, the value for $C \to 0$ and $\rho \to 0.5$. I run the simulation allowing $p \in \{0, 0.5\}$ in increments of 0.1 and $\alpha \in \{0.1, 0.95\}$ in increments of 0.05 to observe what happens to C. I keep $t_F = 200,000$. The results remain the same as before and are shown in Figure 3.4. C_{upper} remains the same as before, but C_{lower} becomes:

Lower Bound:

$$C_{lower} = \frac{3 \times (\langle k \rangle - 1)}{2 \times (2\langle k \rangle - 1)} \times (1-p)^3$$

$$C_{lower} = \frac{3 \times (5-1)}{2 \times (2 \times 5 - 1)} \times (1-0.5)^3$$

$$C_{lower} = \frac{3 \times 4}{2 \times 9} \times (0.5)^3$$

$$C_{lower} = \frac{12}{18} \times (0.125)$$

$$C_{lower} = \frac{2}{3} \times (0.125) = 0.083$$

I then run the simulation allowing $p \in \{0, 0.5\}$ in increment of 0.05 and $\alpha \in \{0.1, 0.9\}$ in increments of 0.1. I also increase t_F to be $t_F = 1,500,000$ and examine how these changes affect my results. The upper and lower bounds for C remain the same as in the previous trial, when $p \in \{0, 0.5\}$ in increments of 0.1 and $\alpha \in \{0.1, 0.95\}$ in increments of 0.05, and the results also remain the same. This is shown in Figure 3.5. One can observe that the critical value C_C also remains as $C_C \approx 0.25$. The case for the jammed state of this simulation is shown

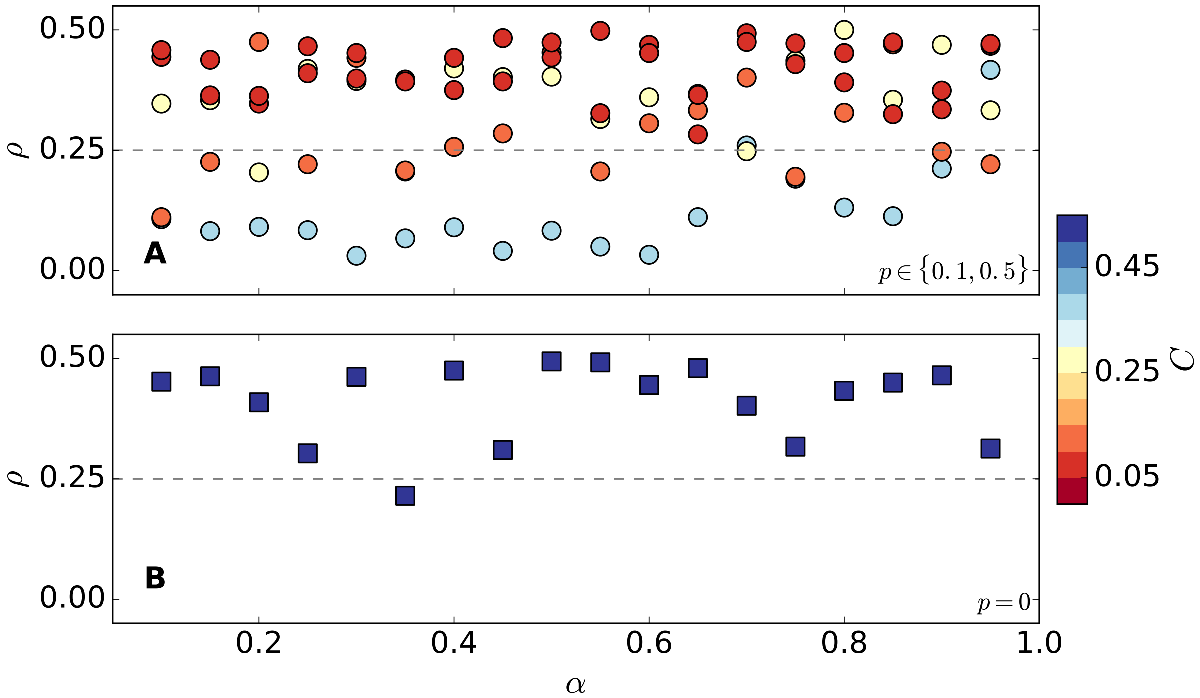

Figure 3.4: Progression of α versus ρ for $N = 1000$, $\langle k \rangle = 5$, $\lambda = \frac{1}{2}$, and $t_F = 200,000$ when $p \in \{0, 0.5\}$ in increments of 0.1 and $\alpha \in \{0.1, 0.95\}$ in increments of 0.05. The colors of the dots represent C. As C decreases, ρ increases except for when $p = 0$ due to the highly clustered nature of the network at that p value. Note there exists slightly more variation in ρ than in Figure 3.2.

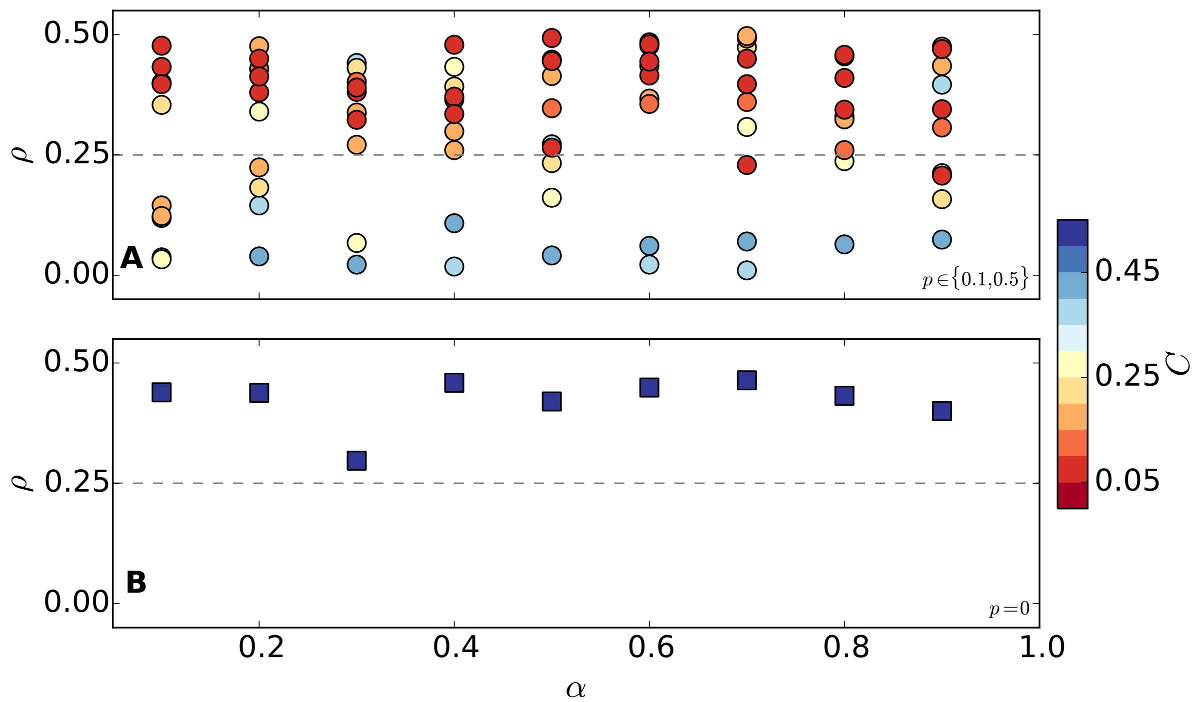

Figure 3.5: Progression of α versus ρ for $N = 1000$, $\langle k \rangle = 5$, $\lambda = \frac{1}{2}$, and $t_F = 1,500,000$ when $p \in \{0, 0.5\}$ in increments of 0.05 and $\alpha \in \{0.1, 0.9\}$ in increments of 0.1. The colors of the dots represent C. As C decreases, ρ increases except for when $p = 0$ due to the highly clustered nature of the network at that p value. Note there exists slightly more variation in ρ than in Figure 3.2.

in Figure 3.6.

One interesting aspect evident from these graphs of α versus ρ is that higher a C yields a lower ρ. We know that social networks tend to show non-trivial clustering— in that context the above observation is very significant. This result indicates that one must take into account the influences of clustering on contagion dynamics on a social network.

My results also further signify that not only does clustering in one's network influence the chance of contracting a contagion, but it also influences the speed at which one can contract a contagion. The jammed states graphs show that as C increases, the time it takes for the simulation to complete, t, decreases for cases that do not result in a jammed state. This means that in highly clustered networks a consensus state is reached faster with less churning in the society. Also, highly clustered networks can show both kinds of consensus, namely unanimous

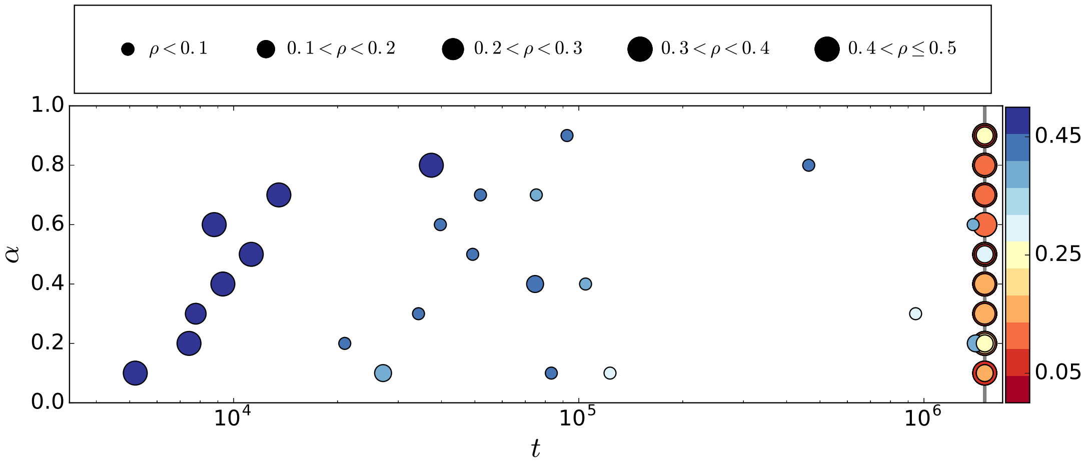

Figure 3.6: Depiction of a jammed state. Progression of t versus α when $N = 1000$, $\langle k \rangle = 5$, $\lambda = \frac{1}{2}$, and $t_F = 1,500,000$. The color of the dots represent C, while the size of the dots represents ρ. Note the logarithmic scale on t and the ticks for t are in powers of 10. The grey line represents the cutoff time, t_F. When $C \leq 0.25$ we have a jammed state. For $0.25 < C < 0.45$ we have ρ values $0 \leq \rho < 0.2$. For $C \geq 0.45$ we have ρ values $0.3 < \rho \leq 0.5$. In addition, higher C yields a shorter simulation time t.

consensus ($\rho = 0$) or fractured consensus ($\rho = 0.5$).

Next I study the temporal trajectories of different variables involved in the model. Figure 3.7 shows plots between N_1 and L_{01} with time, highlighted by color. The parameters I use to generate Figure 3.7(A) are $N = 500$, $p = 0$, $\langle k \rangle = 6$, $\lambda = \frac{1}{2}$, and $\alpha = 0.4$. The parameters I use to generate Figure 3.7(B) are $N = 500$, $p = 1$, $\langle k \rangle = 4$, $\lambda = \frac{1}{3}$, and $\alpha = 0.8$. I observe trajectories which resemble random walks in two dimensional space, though I have not carried out the analysis required to establish the same result. Figure 3.7(A) and (B) show the cases in which I observe convergent dynamics. Observe that in Figure 3.7(A) there will exist a small minority population whereas in Figure 3.7(B) there will exist a large minority population.

What happens when the model results in a nonconvergent, or jammed, state? Figure 3.7(C) demonstrates this case. For this simulation, the parameters I use are $p = 1$, $\langle k \rangle = 4$, $\lambda = \frac{1}{2}$, and $\alpha = 0.6$. I allow the model to run for 200,000 run-throughs, yielding 200,000 data points of N_1 versus L_{01}. It begins with $N_1 = \frac{1}{2}N$ due to the initial distribution of opinions. However, once again we observe that this distribution remains fairly constant as $L_{01} \to 0$. However, L_{01}

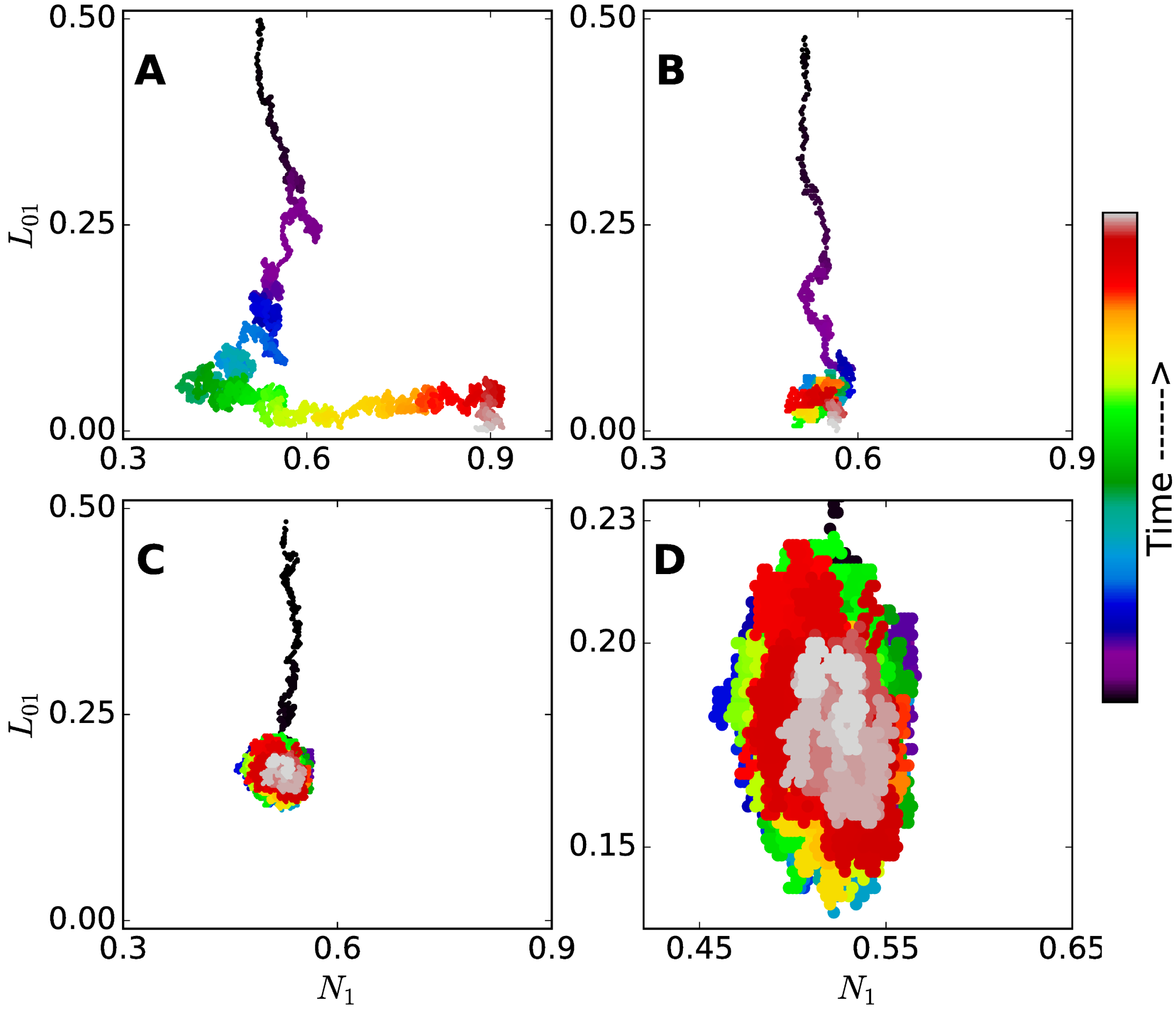

Figure 3.7: The colorbar on the right of the graph shows the progression of time. (A) N_1 versus L_{01} as time increases for $N = 500$ and $\rho = 0$. The initial distribution of nodes' opinions is $N_1 = \frac{1}{2}N$. There is much discord in N_1 as $L_{01} \to 0$, however it finalizes at the distribution of $N_1 = N$. (B) N_1 versus L_{01} as time increases for $N = 500$ and $\rho = 0.5$. The initial distribution of nodes' opinions is $N_1 = \frac{1}{2}N$, and this distribution remains nearly constant until $L_{01} = 0$. (C) N_1 versus L_{01} as time increases for $\lim_{t \to t_F} L_{01}(t) \neq 0$, or a jammed state, for $N = 500$. The initial distribution of nodes' opinions is $N_1 = \frac{1}{2}N$, and this distribution remains nearly constant. Note that L_{01} settles in around the value $L_{01} \approx 0.15L$. (D) A zoomed in version of (C). As time progresses the circles of red overlap and form nearly concentric circles rather than migrating downward to the case where $L_{01} = 0$.

never reaches 0. It reaches around $0.15L$, but then stays in that range as $t \to \infty$. Figure 3.7(D) represents a zoomed in version of Figure 3.7(C). One can observe in the picture that the circle of red, representing time progression, becomes smaller and forms concentric circles rather than migrating downward toward the case where $L_{01} = 0$. Once the network reaches this jammed state, L_{01} and N_1 remain constant.

My above model is simple and minimalistic. It incorporates only a few basic processes that might be involved in opinion formation, but it is still able to reproduce a wide array of real-world scenarios. For example, in light of the current United States presidential election, the recently concluded Kentucky Democratic party primary election ended in nearly a 50%-50% split between Hillary Clinton and Bernie Sanders [19]. This is similar to the case in my simulation when $\rho = 0.5$, the fractured consensus, since the opinions split evenly amongst the nodes of the network. The other extreme, $\rho = 0$, the united consensus, is similar to the Vermont Democratic party primary election where Bernie Sanders defeated Hillary Clinton around 86.1%-13.6% [19]. Although it is not a perfect case, it is closer to the case where there exists a clear majority opinion, versus an almost even split amongst opinions. Such political divide has been a topic of discussion for years, since 2008 when Bill Bishop wrote his book *The Big Sort*. An article from The Washington Post found that Americans have become more clustered according lifestyle and that this geographical clustering has been a source of political divide [23]. Our results mimic this finding, since ρ depends on C.

Chapter 4

Analysis of Facebook data

"Mathematics is the music of reason."

- James Joseph Sylvester

One of my motivations for developing this model is to analyze structural properties of real-world network data sets. Using data from Stanford's Large Network Dataset Collection (SNAP), I run the model on Facebook data consisting of 4,039 nodes and 88,234 edges [11]. Figure 4.1 shows a representation of this network prior to the model being applied. A few key differences between this simulation and my model on the Watts-Strogatz network include lacking directly controllable parameters, p and $\langle k \rangle$, since C and $\langle k \rangle$ are inherent in the Facebook network data. I do, however, control the distribution of O on N and the parameter α. I allow $t_F = 1,500,000$. I first divide the data into $\lambda = \frac{1}{50}$. Interestingly, $\rho \approx 0.5 \;\; \forall \alpha \in \{0.1, 0.9\}$ in increments of 0.1 and $C = 0.51917$. This is analogous to the case where $p = 0$ on the Watts-Strogatz network, since that is the highest value of C achieved for that network. This is shown in Figure 4.2.

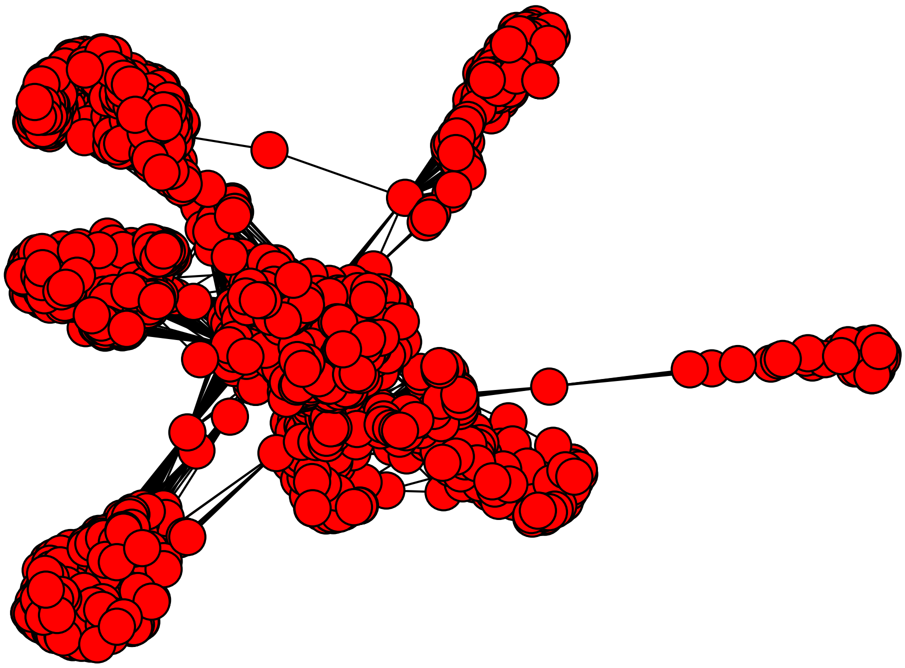

Figure 4.1: Graph of a real-world Facebook network.

In order to generate more interesting results with this simulation, I then try sampling the data before running my model. I let the Facebook data equal G_0 and randomly sample 1000 of the nodes to create a subgraph $G_1 \subset G_0$, before dividing $G_1 \subset G_0$ into further subgraphs, $G_1' \subset G_1 \subset G_0$, where λ represents the division of subgraphs. In this way I test if a smaller number of subdivisions will change C and thus ρ. I repeat this sampling process 5 times, taking a different combination of 1000 nodes each time to examine the effects on C and ρ so the results are not unique to a specific subgroup of nodes. The reason for first sampling and not just running my model on the entire network with fewer subdivisions is that the network is too large. It will take too much time to perform the simulation on such a large network, given that my computer has limited memory space. One way to sample nodes is to use a formal

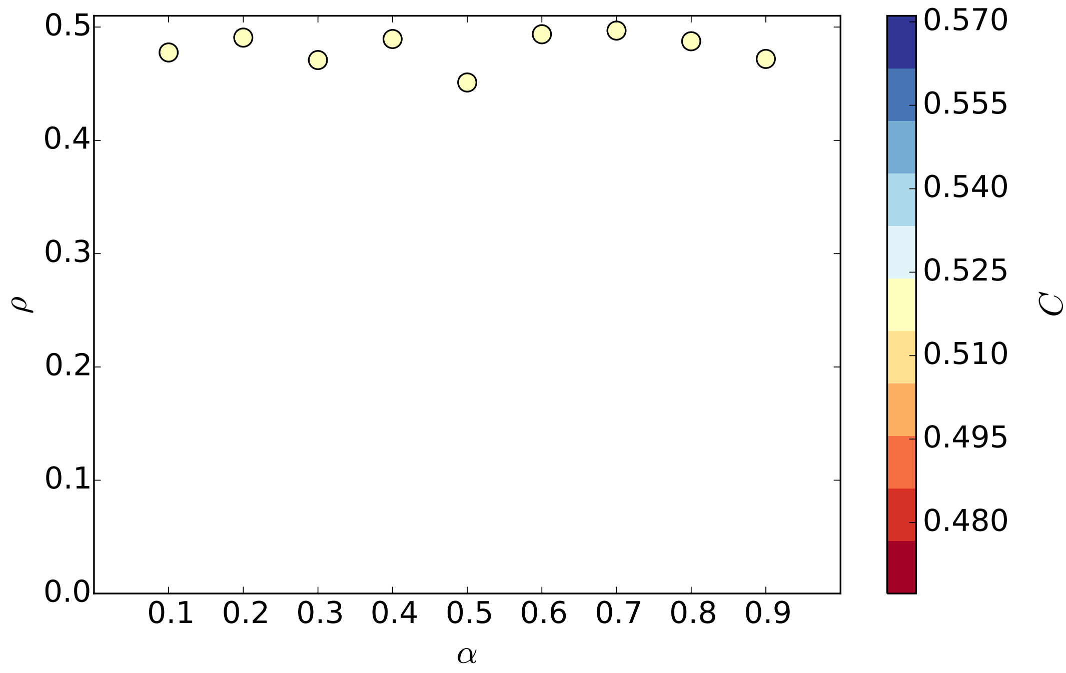

Figure 4.2: α versus ρ for Facebook data, when $\lambda = \frac{1}{50}$. The colors of the dots represent C. There are high values of C and ρ for all α values.

sampling method. Although I did not do that in my research, that would be an area of further investigation.

I test cases where $\lambda = \frac{1}{10}$, $\lambda = \frac{1}{20}$, $\lambda = \frac{1}{30}$, and $\lambda = \frac{1}{40}$. Interestingly, for all cases, $C \approx 0.5$, normally staying in the range $0.4 \leq C \leq 0.6$. $\rho \approx 0.5$ for all cases, but the time t it takes to complete the simulation lessens as λ decreases. Figure 4.3 shows graphs of α versus ρ, where the colors of the dots represents C for different λ values.

The Facebook network is a very dense network with high clustering. In this way, $\rho \approx 0.5$ for most values of C in the Facebook data. This case is similar to that in the Watts-Strogatz network when $p = 0$, yielding a maximum for C. In fact, C for the Facebook data reaches values $C > 0.5$. This may have to do with the fact that my model divides a dense network into many smaller subgraphs, so the actual voter model dynamics do not reach their full effects. My main conclusion from this analysis is that, under a variety of parameter regimes attempted, the

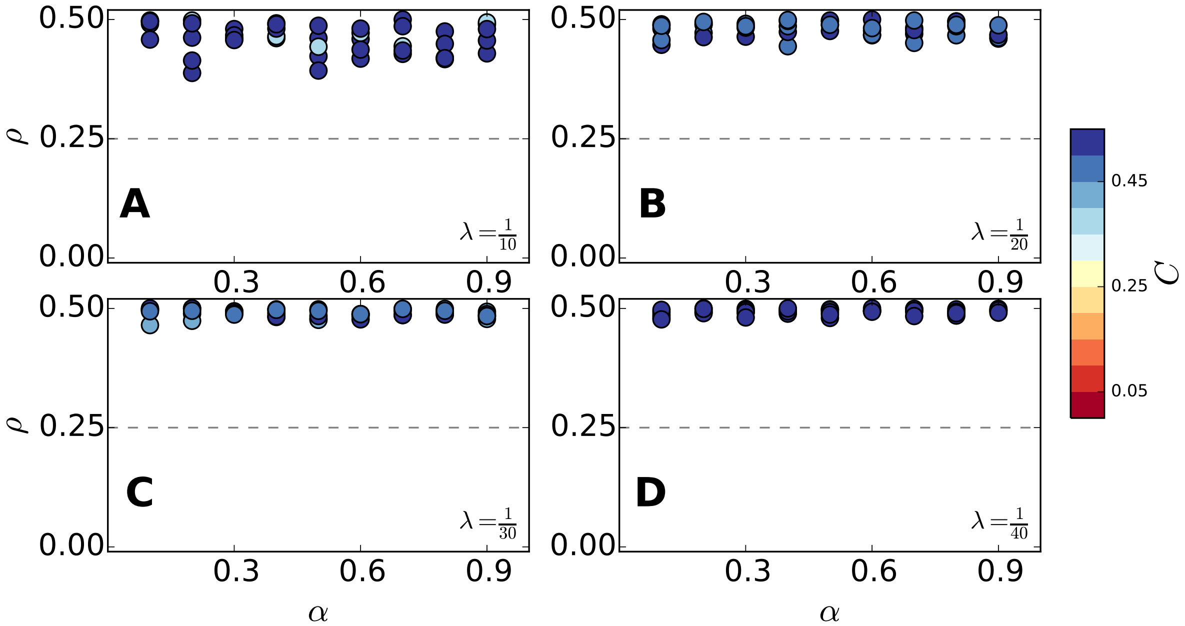

Figure 4.3: α *versus* ρ *for Facebook data with an initial sampling of 1000 nodes and* $t_F = 1,500,000$. *(A) shows the case when* $\lambda = \frac{1}{10}$. *(B) shows the case when* $\lambda = \frac{1}{20}$. *(C) shows the case when* $\lambda = \frac{1}{30}$. *(D) shows the case when* $\lambda = \frac{1}{40}$. *The colors of the dots represent C. There is high clustering C and* ρ *for all values of* α *for all subdivisions* λ. *This is synonymous to when* $p = 0$ *in the Watts-Strogatz network, due to the high density and clustering present in the network.*

Facebook network never undergoes an unanimous consensus, rather it always has a fractured consensus. The reason seems to be the dense nature of the network data and the very high clustering in the data.

Chapter 5

Conclusions

"Probable impossibilities are to be preferred to improbable possibilities."

- Aristotle

I create a model for opinion formation on a static network, employing the Watts-Strogatz network and voter model. I simplify the voter model to have opinions 0 and 1 and examine the structure of the underlying model and how it influences the voter dynamics. I observe that the density of the minority opinion ρ and end states in the model can be fundamentally altered by the clustering C in the model. As C increases, ρ decreases and vice versa. The one exception is when the clustering determinant $p = 0$, since that forms a complete ring topology and consensus is reached before the dynamics can take effect. I observe that there exists a critical value for C, which I identify to be between $C_C \approx 0.25$ for the Watts-Strogatz network, that marks the transition between unanimous consensus and fractured consensus. I also observe that the consensus states are reached faster on clustered networks than on random networks, which is counterintuitive. Another important finding is the existence of a jammed state for certain combinations of $\langle k \rangle$ and p, in which the number of discordant edges never reaches 0 and consensus is never achieved.

In this work I also attempt to develop a technique to use the dynamics on networks for analyzing real-world network data. I analyze Facebook data, which has very high clustering. I determine that the Facebook data is similar to the case when $p = 0$ in a Watts-Strogatz network due to the high clustering of the network and find that it is not possible to achieve a unanimous consensus on Facebook data.

My findings emphasize the importance of the structural properties of networks in the processes involved in collective opinion formation. This work suggests that models which do not explore the role of clustering in the spread of contagions on networks may be of limited applicability.

Appendix A

Python code

A.1 Main model

A.1.1 Functions module

```
# Import all the required packages

import matplotlib.pyplot as plt
import numpy as np
import scipy as sci
from scipy import stats
import networkx as netx
import random as random
from random import choice

# Function for calculating discordant edges

def dis_calcu(G, O):
    edges_in_G=np.array(G.edges())
    D_f=np.abs(O[edges_in_G[:, 0]]-O[edges_in_G[:, 1]])
    Didx_f=np.nonzero(D_f==1)
    discordant_edges_f=edges_in_G[Didx_f]
    return discordant_edges_f

# Function for  finding indices of discordant edges

def dis_idx(G, O):
    edges_in_G=np.array(G.edges())
    D_f=np.abs(O[edges_in_G[:, 0]]-O[edges_in_G[:, 1]])
    Didx_f=np.nonzero(D_f==1)

    return Didx_f

# Function for voter model steps
```

```
def voter_model_step(G0, G1, O, alpha):

    g1edges=np.array(G1.edges())
    g0edges=np.array(G0.edges())

    discordant_edges=dis_calcu(G1, O)

    edge_index=random.randrange(len(discordant_edges));
    nL=discordant_edges[edge_index][0];
    nr=discordant_edges[edge_index][1]

    xi=random.uniform(0,1)

    if xi>alpha:
        O[nL]=O[nr]
    else:
        G01=netx.difference(G0, G1)
        discordant_edges=dis_calcu(G01, O)
        xn1=choice(discordant_edges); nL1=xn1[0]; nr1=xn1[1]

        G1.remove_edge(nL, nr)
        G1.add_edge(nL1, nr1)

    return [G1, G0, O]
```

A.1.2 Main simulation code

```
# import all the required packages

import matplotlib.pyplot as plt
import numpy as np
import scipy as sci
from scipy import stats
import networkx as netx
import random as random
from voter_model_fx import *

avg_degree=5 # average degree k
divide=2 # number of subgraphs, lambda

d1_alpha_rho=np.zeros((5, 99))
k1=0;

# run over range of p0, the clustering determinant

for p0 in np.arange(0.0, 1.1, 1.0):
    no_of_nodes=1000
    G0=netx.watts_strogatz_graph(no_of_nodes, avg_degree, p0)

    # run over range of alpha
```

```
    for alpha in np.arange(0.1, 1.0, 0.1):
        counter=0
        O=np.zeros(no_of_nodes)
        O[0:(no_of_nodes/2)]=1; O[(no_of_nodes/2):no_of_nodes]=0
        np.random.shuffle(O); O=np.int32(O); # evenly distributed opinion vector

        g0edges=np.array(G0.edges())
        ac_n=np.int32(len(g0edges)/divide)
        np.random.shuffle(g0edges)
        active_edges=g0edges[0:ac_n]
        G1=netx.create_empty_copy(G0)
        G1.add_edges_from(active_edges)

        g1edges=np.array(G1.edges())

        discordant_edges=dis_calcu(G0, O)

        discordant_edges=dis_calcu(G1, O)

        while len(discordant_edges)!=0:
            [G1, G0, O]=voter_model_step(G0, G1, O, alpha)
            rho1=sum(O)/np.float32(no_of_nodes);
            discordant_edges=dis_calcu(G1, O)
            if(rho1 >0.5):
                rho=abs(rho1 -1.0)
            else:
                rho=rho1

            g1edges=np.array(G1.edges())
            counter=counter+1

            if(counter >1500000): # run the code until t=t_F
                break;

        d1_alpha_rho[0, k1]=alpha
        d1_alpha_rho[1, k1]=rho
        d1_alpha_rho[2, k1]=p0
        d1_alpha_rho[3, k1]=counter
        d1_alpha_rho[4, k1]=netx.transitivity(G0)
        k1=k1+1
```

A.2 Plotting Figure 2.1, diagram of simulation

```
# import all the required packages

import matplotlib.pyplot as plt
import numpy as np
import scipy as sci
from scipy import stats
import networkx as netx
```

```
import random as random
from voter_model_fx import *


import networkx as nx
from networkx.drawing.nx_agraph import graphviz_layout

#for rho=0, use p=0, avg_deg=6, divide=2, alpha=0.4
#for rho=0.5, use p=1, avg_deg=4, divide=3, alpha=0.8
#for when L01 does not converge to 0, use p=1, avg_deg=4, divide=2, alpha
    =0.6

avg_degree=4
divide=3

p0=1
alpha=0.8

d1_alpha_rho=np.zeros((3, 99))
k1=0;
no_of_nodes=500

G0=netx.watts_strogatz_graph(no_of_nodes, avg_degree, p0)
O=np.zeros(no_of_nodes)
O[0:(no_of_nodes/2)]=1; O[(no_of_nodes/2):no_of_nodes]=0
np.random.shuffle(O); O=np.int32(O);

g0edges=np.array(G0.edges())
ac_n=np.int32(len(g0edges)/divide)
np.random.shuffle(g0edges)
active_edges=g0edges[0:ac_n]
c=np.zeros(len(g0edges))
c[active_edges]=1

# create edges' colors based on harmonious/discordant

cx=np.ones((len(g0edges), 3))
cx=cx*[0.75, 0.75, 0.75]

ncx=np.zeros((len(O), 3))
ncx[O==1]=[1.0, 0.0, 0.0] #red is opinion 1
ncx[O==0]=[0.0, 0.0, 1.0]  #blue is opinion 0
plt.figure(figsize=(12, 12))

# draw the original network, G_0

pos=graphviz_layout(G0,prog='neato')
plt.axis('off')
nodes = netx.draw_networkx_nodes(G0, pos, node_size=120, node_color=ncx)
nodes.set_edgecolor('none')

netx.draw_networkx_edges(G0, pos, edge_color=cx, width=1.0)

plt.savefig('drawing_original_G0_rho=0.0.eps')
plt.show()
```

```
G1=netx.create_empty_copy(G0)
G1.add_edges_from(active_edges)

g1edges=np.array(G1.edges())

discordant_idx=dis_idx(G1, O)

c=np.zeros(len(g1edges))
c[discordant_idx]=1

cx=np.zeros((len(g1edges), 3))
cx[c==1]=[1.0, 0.457, 0.0] # discordant edges are orange
cx[c==0]= [0.3, 0.7, 0.3] # harmonious edges are green

ncx=np.zeros((len(O), 3))
ncx[O==1]=[1.0, 0.0, 0.0] # red is opinion 1
ncx[O==0]=[0.0, 0.0, 1.0]  # blue is opinion 0

plt.figure(figsize=(12, 12))

# draw the original subgraph, G_1

pos=graphviz_layout(G1,prog='neato')
plt.axis('off')
nodes = netx.draw_networkx_nodes(G1, pos, node_size=120, node_color=ncx)
nodes.set_edgecolor('none')

netx.draw_networkx_edges(G1, pos, edge_color=cx, width=1.0)

discordant_edges=dis_calcu(G1, O)

plt.savefig('drawing_original_G1_rho=0.0.eps')
plt.show()

counter=0
while len(discordant_edges)!=0:
    [G1, G0, O]=voter_model_step(G0, G1, O, alpha)
    rho1=sum(O)/np.float32(no_of_nodes);
    discordant_edges=dis_calcu(G1, O)
    if(rho1 >0.5):
        rho=abs(rho1 -1.0)
    else:
        rho=rho1

    g1edges=np.array(G1.edges())

    print rho1, rho, alpha, sum(O), len(discordant_edges)
print alpha, rho, sum(O)

c=np.zeros(len(g1edges))
discordant_idx=dis_idx(G1, O)

c[discordant_idx]=1
```

```
cx=np.zeros((len(g1edges), 3))
cx[c==1]=[1.0, 0.457, 0.0] # discordant edges are orange
cx[c==0]= [0.3, 0.7, 0.3] # harmonious edges are green

ncx=np.zeros((len(O), 3))
ncx[O==1]=[1.0, 0.0, 0.0] #red is opinion 1
ncx[O==0]=[0.0, 0.0, 1.0]  #blue is opinion 0

plt.figure(figsize=(12, 12))

# draw the new subgraph, G_1, with no discordant edges

os=graphviz_layout(G1, prog='neato')
plt.axis('off')
nodes = netx.draw_networkx_nodes(G1, pos, node_size=120, node_color=ncx)
nodes.set_edgecolor('none')
netx.draw_networkx_edges(G1, pos, edge_color=cx, width=1.0)

plt.savefig('drawing_rho=0.0.eps')
plt.show()

d1_alpha_rho[0, k1]=alpha
d1_alpha_rho[1, k1]=rho
d1_alpha_rho[2, k1]=p0

print alpha, rho, sum(O)
k1=k1+1
```

A.3 Plotting Figure 3.1, C for my model

A.3.1 Find C for G_0

```
# import all the required packages

import matplotlib.pyplot as plt
import numpy as np
import scipy as sci
from scipy import stats
import networkx as netx
import random as random
from voter_model_fx import *

avg_degree=5
divide=2
d1_alpha_rho=np.zeros((3, 11))
k1=0;

# calculate transitivity for the whole range of clustering determinant, p0

for p0 in np.arange(0.0, 1.1, 0.1):
    no_of_nodes=5000
```

```
    G0=netx.watts_strogatz_graph(no_of_nodes, avg_degree, p0)

    d1_alpha_rho[0, k1]=netx.transitivity(G0)
    d1_alpha_rho[1, k1]=netx.average_shortest_path_length(G0)
    d1_alpha_rho[2, k1]=p0
    print d1_alpha_rho[0, k1], d1_alpha_rho[1, k1], d1_alpha_rho[2, k1]
    k1=k1+1

np.savetxt('CvsL.txt', d1_alpha_rho)
```

A.3.2 Plot the graph of C

```
# import all the required packages

import matplotlib.pyplot as plt
import numpy as np
import scipy as sci
from scipy import stats
import networkx as netx
import random as random
import math

d1_alpha_rho=np.loadtxt('CvsL.txt')

# p is the list of p0 values

p=[]
for i in range (1, len(d1_alpha_rho[2, :])):
    q=d1_alpha_rho[2, i]
    p.append(q)
print p

# plot C/C0 for when p=0

plt.scatter(0, (d1_alpha_rho[0, 0])/d1_alpha_rho[0, 0], s=120, c='b',
    marker='o')

# plot C/C0 for when p!=0

plt.scatter(p, (d1_alpha_rho[0, 1:])/d1_alpha_rho[0, 0], s=120, c='b',
    marker='o')
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.ylim(0, 1.2)
plt.xlabel('$p$', labelpad=10, fontsize=24)
plt.ylabel(r'$\frac{C_p}{C_0}$', fontsize=24)
plt.tight_layout()
plt.savefig('CvsL_mine.eps')
plt.show()
```

A.4 Plotting α versus ρ for when $t_F = 200,000$

```
#import all the required packages

import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
from time import time
import matplotlib as mpl

# set the standard parameters

params = { 'figure.figsize': (10, 6),
     'axes.labelsize': 20,
          'text.fontsize': 24,
               'xtick.labelsize': 18,
                    'ytick.labelsize': 18,
                         'legend.fontsize': 18,
                              'text.usetex': False,
                                   'mathtext.bf': 'helvetica:bold',
                         }

plt.rcParams.update(params)

# create a colorbar

col01=[165./255., 0./255., 38./255.]
col1=[215./255., 48./255., 39./255.]
col2=[244./255., 109./255., 67./255.]
col3=[253./255., 174./255., 97./255.]
col4=[254./255., 224./255., 144./255.]
col41=[255./255., 255./255., 191./255.]
col42=[224./255., 243./255., 248./255.]
col51=[171./255., 217./255., 233./255.]
col52=[116./255., 173./255., 209./255.]
col6=[69./255., 117./255., 180./255.]
col61=[49./255., 54./255.,1 49./255.]

col0=np.array([col01, col1, col2, col3, col4, col41, col42, col51, col52,
    col6, col61])

cm = mpl.colors.ListedColormap(col0)

# load the necessary data

d1_alpha_rho=np.loadtxt('1000timed_k5_acn2_counter200000.txt')

f, axes = plt.subplots(2, 2)

# plot alpha versus rho for when p0!=0

ax1=plt.subplot(2, 1, 1)

ax1.axhline(y=.25, xmin=0.0, xmax=1.0, ls='--', color='grey')
```

```
ax1.scatter(d1_alpha_rho[0, 9:], d1_alpha_rho[1, 9:], s=120, c=d1_alpha_rho
    [4, 9:], cmap=cm,marker='o', vmin=0, vmax=0.55)
ax1.annotate('$p$' r'$\in \{0.1,1.0\}$', xy=(1, 0.01), xycoords='axes
    fraction', fontsize=15, xytext=(-5, 5), textcoords='offset points', ha='
    right', va='bottom')

ax1.set_xticks([0.2, 0.4, 0.6, 0.8, 1.0])
ax1.set_yticks([0.0, 0.25, 0.5])
ax1.set_ylabel(r'$\rho$', fontsize=20)
ax1.set_xticklabels([])
ax1.set_xlim([0.05, 1])
ax1.set_ylim([-0.05, 0.55])

# plot alpha versus rho for when p0=0

ax2=plt.subplot(2, 1, 2)

ax2.axhline(y=.25, xmin=0.0, xmax=1.0, ls='--', color='grey')
p1=ax2.scatter(d1_alpha_rho[0, 0:9], d1_alpha_rho[1, 0:9], s=120, c=
    d1_alpha_rho[4, 0:9], cmap=cm, marker='s', vmin=0, vmax=0.55)
ax2.annotate('$p=0$', xy=(1, 0.01), xycoords='axes fraction', fontsize=15,
    xytext=(-5, 5), textcoords='offset points', ha='right', va='bottom')

# format the axes

ax2.set_xticks([0.2, 0.4, 0.6, 0.8, 1.0])
ax2.set_yticks([0.0, 0.25, 0.5])
ax2.set_xlim([0.05, 1])
ax2.set_ylim([-0.05, 0.55])
ax2.set_ylabel(r'$\rho$', fontsize=20)
ax2.set_xlabel(r'$\alpha$', fontsize=20)
ax1.set_position([0.1, 0.575, 0.77, 0.4])
ax2.set_position([0.1, 0.12, 0.77, 0.4])

# add text to label subplots

f.text(0.145, 0.2, 'B', transform=ax2.transAxes, fontsize=18, fontweight='
    bold', va='top', ha='right')
f.text(0.145, 0.65, 'A', transform=ax1.transAxes, fontsize=18, fontweight='
    bold', va='top', ha='right')

# add the colorbar

cbar_ax = f.add_axes([0.885, 0.285, 0.025, 0.4])
cbar=f.colorbar(p1, cax=cbar_ax,ticks=([0.05, 0.25, 0.45]))
cbar.set_label(r'$C$', fontsize=20)

plt.savefig('1000timed_k5_acn2_counter200000_new.eps')
plt.show()
```

A.5 Plotting Figure 3.5, plotting α versus ρ for when $t_F = 1,500,000$

```
#import all the required packages

import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
from time import time
import matplotlib as mpl

# set the standard parameters

params = { 'figure.figsize': (10, 6),
    'axes.labelsize': 20,
        'text.fontsize': 24,
            'xtick.labelsize': 18,
                'ytick.labelsize': 18,
                    'legend.fontsize': 18,
                        'text.usetex': False,
                            'mathtext.bf': 'helvetica:bold',
                    }

plt.rcParams.update(params)

#create a colorbar

col01=[165./255., 0./255., 38./255.]
col1=[215./255., 48./255., 39./255.]
col2=[244./255., 109./255., 67./255.]
col3=[253./255., 174./255., 97./255.]
col4=[254./255., 224./255., 144./255.]
col41=[255./255., 255./255., 191./255.]
col42=[224./255., 243./255., 248./255.]
col51=[171./255., 217./255., 233./255.]
col52=[116./255., 173./255., 209./255.]
col6=[69./255., 117./255., 180./255.]
col61=[49./255., 54./255., 149./255.]

col0=np.array([col01, col1, col2, col3, col4, col41, col42, col51, col52,
    col6, col61])

cm = mpl.colors.ListedColormap(col0)

# load the necessary data

data=np.genfromtxt('timed_data_anm_brk.txt')

f, axes = plt.subplots(2, 2)

# plot alpha versus rho for when p0!=0

ax1=plt.subplot(2, 1, 1)

```

```
ax1.axhline(y=.25, xmin=0.0, xmax=1.0, ls='--', color='grey')
ax1.scatter(data[9:, 1], data[9:, 3], s=120, c=data[9:, 2], cmap=cm, marker
    ='o', vmin=0, vmax=0.55)
ax1.annotate('$p$' r'$\in \{0.1,0.5\}$', xy=(1, 0), xycoords='axes fraction
    ', fontsize=12, xytext=(-5, 5), textcoords='offset points', ha='right',
    va='bottom')

# format the axes

ax1.set_xticks([0.2, 0.4, 0.6, 0.8, 1.0])
ax1.set_yticks([0.0, 0.25, 0.5])
ax1.set_ylabel(r'$\rho$', fontsize=20)
ax1.set_xticklabels([])
ax1.set_xlim([0.05, 1])
ax1.set_ylim([-0.05, 0.55])

# plot alpha versus rho for p0=0

ax2=plt.subplot(2, 1, 2)

ax2.axhline(y=.25, xmin=0.0, xmax=1.0, ls='--', color='grey')
p1=ax2.scatter(data[0:9, 1], data[0:9, 3], s=120, c=data[0:9, 2], cmap=cm,
    marker='s', vmin=0, vmax=0.55)
ax2.annotate('$p=0$', xy=(1, 0), xycoords='axes fraction', fontsize=12,
    xytext=(-5, 5), textcoords='offset points', ha='right', va='bottom')

# format the axes

ax2.set_xticks([0.2, 0.4, 0.6, 0.8, 1.0])
ax2.set_yticks([0.0, 0.25, 0.5])
ax2.set_xlim([0.05, 1])
ax2.set_ylim([-0.05, 0.55])
ax2.set_ylabel(r'$\rho$', fontsize=20)
ax2.set_xlabel(r'$\alpha$', fontsize=20)
ax1.set_position([0.1, 0.575, 0.77, 0.4])
ax2.set_position([0.1, 0.12, 0.77, 0.4])

# add text to label subplots

f.text(0.125, 0.2, 'B', transform=ax2.transAxes, fontsize=18, fontweight='
    bold', va='top', ha='right')
f.text(0.125, 0.65, 'A', transform=ax1.transAxes, fontsize=18, fontweight='
    bold', va='top', ha='right')

# add the colorbar

cbar_ax = f.add_axes([0.885, 0.285, 0.025, 0.4])
cbar=f.colorbar(p1, cax=cbar_ax, ticks=([0.05,0.25,0.45]))
cbar.set_label(r'$C$',fontsize=20)

plt.savefig('tF1500000_interval.eps')
plt.show()
```

A.6 Plotting Figure 3.3 and 3.6, representation of a jammed state

```
# import all the required packages

import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
from time import time
import matplotlib as mpl

# set the standard parameters

params = { 'figure.figsize': (14, 5.5),
    'axes.labelsize': 20,
        'text.fontsize': 24,
            'xtick.labelsize': 18,
                'ytick.labelsize': 18,
                    'legend.fontsize': 18,
                        'text.usetex': False,
                            'mathtext.bf': 'helvetica:bold',
                    }

plt.rcParams.update(params)

# create a colorbar

col01=[165./255., 0./255., 38./255.]
col1=[215./255., 48./255.,39./255.]
col2=[244./255., 109./255., 67./255.]
col3=[253./255., 174./255., 97./255.]
col4=[254./255., 224./255., 144./255.]
col41=[255./255., 255./255., 191./255.]
col42=[224./255., 243./255., 248./255.]
col51=[171./255., 217./255., 233./255.]
col52=[116./255., 173./255., 209./255.]
col6=[69./255., 117./255., 180./255.]
col61=[49./255., 54./255., 149./255.]


fig = plt.figure()

col0=np.array([col01, col1, col2, col3, col4, col41, col42, col51, col52,
    col6, col61])

cm = mpl.colors.ListedColormap(col0)

# load the necessary data

d1_alpha_rho=np.loadtxt('timed_data_anm_brk.txt')

# plot t versus alpha
# size of the marker is the value of rho

```

```
plt.axvline(x=1500000, ymin=0.0, ymax=1.0, ls='-', color='grey', lw=3,
    zorder=1)

#type V marker

dx1=d1_alpha_rho[(d1_alpha_rho[:, 3]>0.4) & (d1_alpha_rho[:, 3]<0.51), 6]
dx2=d1_alpha_rho[(d1_alpha_rho[:, 3]>0.4) & (d1_alpha_rho[:, 3]<0.51), 1]
dx3=d1_alpha_rho[(d1_alpha_rho[:, 3]>0.4) & (d1_alpha_rho[:, 3]<0.51), 2]
dx4=d1_alpha_rho[(d1_alpha_rho[:, 3]>0.4) & (d1_alpha_rho[:, 3]<0.51), 3]
p5=plt.scatter(dx1, dx2, s=400.0, c=dx3, cmap=cm, marker='o', alpha=0.95,
    vmin=0.0, vmax=0.5, zorder=2)

#type IV marker

dx1=d1_alpha_rho[(d1_alpha_rho[:, 3]>0.3) & (d1_alpha_rho[:, 3]<0.4), 6]
dx2=d1_alpha_rho[(d1_alpha_rho[:, 3]>0.3) & (d1_alpha_rho[:, 3]<0.4), 1]
dx3=d1_alpha_rho[(d1_alpha_rho[:, 3]>0.3) & (d1_alpha_rho[:, 3]<0.4), 2]
dx4=d1_alpha_rho[(d1_alpha_rho[:, 3]>0.3) & (d1_alpha_rho[:, 3]<0.4), 3]
p4=plt.scatter(dx1, dx2, s=400.0, c=dx3, cmap=cm, marker='o', alpha=0.95,
    vmin=0.0, vmax=0.5, zorder=2)

#type III marker

dx1=d1_alpha_rho[(d1_alpha_rho[:, 3]>0.2) & (d1_alpha_rho[:, 3]<0.3), 6]
dx2=d1_alpha_rho[(d1_alpha_rho[:, 3]>0.2) & (d1_alpha_rho[:, 3]<0.3), 1]
dx3=d1_alpha_rho[(d1_alpha_rho[:, 3]>0.2) & (d1_alpha_rho[:, 3]<0.3), 2]
dx4=d1_alpha_rho[(d1_alpha_rho[:, 3]>0.2) & (d1_alpha_rho[:, 3]<0.3), 3]
p3=plt.scatter(dx1, dx2, s=300.0, c=dx3, cmap=cm, marker='o', alpha=0.95,
    vmin=0.0, vmax=0.5, zorder=2)

#type II marker

dx1=d1_alpha_rho[(d1_alpha_rho[:, 3]>0.1) & (d1_alpha_rho[:, 3]<0.2), 6]
dx2=d1_alpha_rho[(d1_alpha_rho[:, 3]>0.1) & (d1_alpha_rho[:, 3]<0.2), 1]
dx3=d1_alpha_rho[(d1_alpha_rho[:, 3]>0.1) & (d1_alpha_rho[:, 3]<0.2), 2]
dx4=d1_alpha_rho[(d1_alpha_rho[:, 3]>0.1) & (d1_alpha_rho[:, 3]<0.2), 3]
p2=plt.scatter(dx1, dx2, s=200.0, c=dx3, cmap=cm, marker='o', alpha=0.95,
    vmin=0.0, vmax=0.5, zorder=2)

#type I marker

dx1=d1_alpha_rho[(d1_alpha_rho[:, 3]>=0.0) & (d1_alpha_rho[:, 3]<0.1), 6]
dx2=d1_alpha_rho[(d1_alpha_rho[:, 3]>=0.0) & (d1_alpha_rho[:, 3]<0.1), 1]
dx3=d1_alpha_rho[(d1_alpha_rho[:, 3]>=0.0) & (d1_alpha_rho[:, 3]<0.1), 2]
dx4=d1_alpha_rho[(d1_alpha_rho[:, 3]>=0.0) & (d1_alpha_rho[:, 3]<0.1), 3]
p1=plt.scatter(dx1, dx2, s=100.0, c=dx3, cmap=cm, marker='o', alpha=0.95,
    vmin=0.0, vmax=0.5, zorder=2)

# plot a legend for rho

labels = [r'$\rho<0.1$', r'$0.1<\rho<0.2$', r'$0.2<\rho<0.3$', r'$0.3<\rho
    <0.4$', r'$0.4<\rho \leq 0.5$']

leg = plt.legend([p1, p2, p3, p4, p5], labels, ncol=5, bbox_to_anchor
    =(-0.005, 1.01), frameon=True, fontsize=16, handlelength=2, loc =3,
```

```
    borderpad =1.8, handletextpad=0.1, scatterpoints=1)

# format the axes

plt.yticks([0, 0.2, 0.4, 0.6, 0.8, 1])
plt.xscale('log')
plt.ylim(0, 1)
plt.xlim(-100000, 1690000.)
plt.xlabel(r'$t$', fontsize=24)
plt.ylabel(r'$\alpha$', fontsize=24)

# add the colorbar

cbaxes = fig.add_axes([0.9025, 0.125, 0.02, 0.625])
colorbar=plt.colorbar(cax = cbaxes)
colorbar.ax.set_ylabel(r'$C$', rotation=90)
colorbar.set_clim([0.01, 0.5])
colorbar.set_ticks([0.05, 0.25, 0.45])

# format the legend

leg.legendHandles[0].set_color('k')
leg.legendHandles[1].set_color('k')
leg.legendHandles[2].set_color('k')
leg.legendHandles[3].set_color('k')
leg.legendHandles[4].set_color('k')

plt.savefig('jammed_state_plot.eps')
plt.show()
```

A.7 Plotting Figure 3.7, subplot of different convergent states

```
# import all the required packages

import matplotlib.pyplot as plt
import numpy as np
import scipy as sci
from scipy import stats
import networkx as netx
import random as random
import matplotlib

# set the standard parameters

params = { 'figure.figsize': (10, 10),
    'axes.labelsize': 20,
        'text.fontsize': 24,
            'xtick.labelsize': 18,
                'ytick.labelsize': 18,
                    'legend.fontsize': 18,
                        'text.usetex': False,
```

```
                                'mathtext.bf': 'helvetica:bold',
                  }

plt.rcParams.update(params)

# import the necessary data for each case, when rho=0, rho=0.5, and the
    jammed state

SumO_0=np.genfromtxt('L01_sumO_data_p=0.0_normalized.txt', 'f8', usecols=0)
L01_0=np.genfromtxt('L01_sumO_data_p=0.0_normalized.txt', 'f8', usecols=1)

SumO_1=np.genfromtxt('L01_sumO_data_p=0.5_normalized.txt', 'f8', usecols=0)
L01_1=np.genfromtxt('L01_sumO_data_p=0.5_normalized.txt', 'f8', usecols=1)

SumO_2=np.genfromtxt('L01_sumO_data_p=1.0_noconverge_normalized.txt', 'f8',
      usecols=0)
L01_2=np.genfromtxt('L01_sumO_data_p=1.0_noconverge_normalized.txt', 'f8',
    usecols=1)

SumO_3=np.genfromtxt('L01_sumO_data_p=1.0_noconverge_normalized.txt', 'f8',
      usecols=0)
L01_3=np.genfromtxt('L01_sumO_data_p=1.0_noconverge_normalized.txt', 'f8',
    usecols=1)

# color is according to time

t1_0=np.arange(0, len(SumO_0));
t1_0=t1_0/np.float32(len(SumO_0))

t1_1=np.arange(0, len(SumO_1));
t1_1=t1_1/np.float32(len(SumO_1))

t1_2=np.arange(0, len(SumO_2));
t1_2=t1_2/np.float32(len(SumO_2))

t1_3=np.arange(0, len(SumO_2));
t1_3=t1_2/np.float32(len(SumO_2))

f, axes = plt.subplots(2, 2)

# format the plot

super_axis = f.add_subplot(111)
super_axis.set_axis_bgcolor('none')
super_axis.axes.get_xaxis().set_ticks([])
super_axis.axes.get_yaxis().set_ticks([])
super_axis.tick_params(labelcolor='none', top='off', bottom='off', left='
    off', right='off')
super_axis.spines['bottom'].set_color('none')
super_axis.spines['top'].set_color('none')
super_axis.spines['left'].set_color('none')
super_axis.spines['right'].set_color('none')

# plot the data
```

```
axes[0, 0].scatter(SumO_0, L01_0, c=t1_0, marker='.', s=30, edgecolor='none', cmap=plt.cm.nipy_spectral)
axes[0, 0].text(0.1, 0.9, 'A', transform=axes[0, 0].transAxes, fontsize=24, fontweight='bold', va='top', ha='right')
axes[0, 0].set_ylabel(r'$L_{01}$')

axes[0, 1].scatter(SumO_1, L01_1, c=t1_1, marker='.', s=30, edgecolor='none', cmap=plt.cm.nipy_spectral)
axes[0, 1].text(0.1, 0.9, 'B', transform=axes[0, 1].transAxes, fontsize=24, fontweight='bold', va='top', ha='right')

axes[1, 0].scatter(SumO_2, L01_2, c=t1_2, marker='.', s=30, edgecolor='none', cmap=plt.cm.nipy_spectral)
axes[1, 0].text(0.1, 0.9, 'C', transform=axes[1, 0].transAxes, fontsize=24, fontweight='bold', va='top', ha='right')
axes[1, 0].set_xlabel(r'$N_1$')
axes[1, 0].set_ylabel(r'$L_{01}$')

p1=axes[1, 1].scatter(SumO_3, L01_3, c=t1_3, marker='.', s=210, edgecolor='none', cmap=plt.cm.nipy_spectral)
axes[1, 1].text(0.1, 0.9, 'D', transform=axes[1, 1].transAxes, fontsize=24, fontweight='bold', va='top', ha='right')
axes[1, 1].set_xlabel(r'$N_1$')

axes[0, 0].set_position([0.1, 0.5, 0.35, 0.35])
axes[0, 1].set_position([0.525, 0.5, 0.35, 0.35])

axes[1, 0].set_position([0.1, 0.1, 0.35, 0.35])
axes[1, 1].set_position([0.525, 0.1, 0.35, 0.35])

# format the axes

plt.sca(axes[0, 0])
plt.xticks([0.3, 0.6, 0.9])
plt.sca(axes[0, 0])
plt.yticks([0.0, 0.25, 0.5])
plt.ylim(-0.01, 0.51)

plt.sca(axes[0, 1])
plt.xticks([0.3, 0.6, 0.9])
plt.sca(axes[0, 1])
plt.yticks([0.0, 0.25, 0.5])
plt.ylim(-0.01, 0.51)

plt.sca(axes[1, 0])
plt.xticks([0.3, 0.6, 0.9])
plt.sca(axes[1, 0])
plt.yticks([0.0, 0.25, 0.5])
plt.ylim(-0.01, 0.51)

plt.sca(axes[1, 1])
plt.xticks([0.45, 0.55, 0.65])
plt.sca(axes[1, 1])
plt.yticks([0.15, 0.2, 0.23])
```

```
plt.ylim(0.13, 0.235)

# add the colorbar

cbar_ax = f.add_axes([0.9, 0.285, 0.028, 0.4])
cbar=f.colorbar(p1, cax=cbar_ax, ticks=([]), cmap=plt.cm.RdYlGn)
cbar.set_label(r'Time ------->', fontsize=20)

plt.savefig('subplot.eps')
plt.show()
```

A.8 Simulation code for Facebook network

```
# import all the required packages

import matplotlib.pyplot as plt
import numpy as np
import scipy as sci
from scipy import stats
import networkx as netx
import random as random
import math
from voter_model_fx_facebook import *

# load the necessary data

facebook=np.genfromtxt('facebook_data.txt', dtype=[('i8'), ('i8')])
d1_alpha_rho=np.zeros((4, 45))
k1=0;
n_divide=1000 # how many nodes you will be taking in the initial sampling
divide=15 # how many subgraphs the SAMPLING will have

# create a graph from the Facebook data

F=netx.Graph()
F.add_edges_from(facebook)
edges=np.array(F.edges())
nodes=np.array(F.nodes()) # 4039 total nodes

# repeat the sampling process 5 times

for i in range(0, 5):

    # create G_0 from a sampling of nodes from the Facebook data

    np.random.shuffle(nodes)
    n_list=nodes[0:n_divide]
    G0_prime=F.subgraph(n_list)
    mapping=dict(zip(G0_prime.nodes(), range(0, n_divide)))
```

```
    #relabel the nodes in G_0 to be within the range of [0, n_divide]

    G0=netx.relabel_nodes(G0_prime, mapping)
    g0nodes=np.array(G0.nodes())

    # test cases for the entire range of alphas

    for alpha in np.arange(0.1, 1.0, 0.1):
        no_of_nodes=len(g0nodes)
        print no_of_nodes
        counter=0
        O=np.zeros(no_of_nodes)
        O[0:(no_of_nodes/2)]=1; O[(no_of_nodes/2):no_of_nodes]=0
        np.random.shuffle(O); O=np.int32(O);
        print sum(O)

        g0edges=np.array(G0.edges())
        ac_n=np.int32(len(g0edges)/divide)

        # create subgraph G_1 from G_0

        np.random.shuffle(g0edges)
        active_edges=g0edges[0:ac_n]
        G1=netx.create_empty_copy(G0)
        G1.add_edges_from(active_edges)

        g1edges=np.array(G1.edges())

        discordant_edges=dis_calcu(G1,O)

        print sum(O)/np.float32(no_of_nodes), len(g1edges), len(g0edges)

        while len(discordant_edges)!=0:
            [G1, G0, O]=voter_model_step(G0, G1, O, alpha)
            rho1=sum(O)/np.float32(no_of_nodes);
            discordant_edges=dis_calcu(G1, O)
            if(rho1 >0.5):
                rho=abs(rho1 -1.0)
            else:
                rho=rho1

            g1edges=np.array(G1.edges())
            counter=counter+1

            if(counter >1500000): # run the code until t=t_F
                break;

        d1_alpha_rho[0, k1]=alpha
        d1_alpha_rho[1, k1]=rho
        d1_alpha_rho[2, k1]=counter
        d1_alpha_rho[3, k1]=netx.transitivity(G0)

        print alpha, rho, sum(O), len(g1edges), d1_alpha_rho[3, k1],
   counter
```

```
        k1=k1+1

np.savetxt('facebook_subgraph_divide15.txt', d1_alpha_rho)
plt.plot(d1_alpha_rho[0, :], d1_alpha_rho[1, :], 'bo')
plt.xlabel(r'$\alpha$')
plt.ylabel(r'$\rho$')
plt.savefig('facebook_data_subgraph_plot.eps')
plt.title('Facebook Data')
plt.show()
```

A.9 Plotting Figure 4.2, α versus ρ for Facebook data when $\lambda = \frac{1}{50}$

```
#import all the required packages

import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
from time import time
import matplotlib as mpl

# set the standard parameters

params = { 'figure.figsize': (10, 6),
    'axes.labelsize': 20,
        'text.fontsize': 24,
            'xtick.labelsize': 18,
                'ytick.labelsize': 18,
                    'legend.fontsize': 18,
                        'text.usetex': False,
                            'mathtext.bf': 'helvetica:bold',
                    }

plt.rcParams.update(params)

# create a colorbar

col01=[165./255., 0./255., 38./255.]
col1=[215./255., 48./255., 39./255.]
col2=[244./255., 109./255., 67./255.]
col3=[253./255., 174./255., 97./255.]
col4=[254./255., 224./255., 144./255.]
col41=[255./255., 255./255., 191./255.]
col42=[224./255., 243./255., 248./255.]
col51=[171./255., 217./255., 233./255.]
col52=[116./255., 173./255., 209./255.]
col6=[69./255., 117./255., 180./255.]
col61=[49./255., 54./255., 149./255.]

col0=np.array([col01, col1, col2, col3, col4, col41, col42, col51, col52,
    col6, col61])
```

```
cm = mpl.colors.ListedColormap(col0)

# load the necessary data

d1_alpha_rho=np.loadtxt('facebook1_50.txt')

# plot alpha versus rho

plt.scatter(d1_alpha_rho[0, :], d1_alpha_rho[1, :], s=120, c=d1_alpha_rho
    [3, :], cmap=cm, marker='o')
ax.annotate(r'$\lambda= \frac{1}{50}$', xy=(1, 0.1), xycoords='axes
    fraction', fontsize=15, xytext=(-5, 5), textcoords='offset points', ha='
    right', va='bottom')
plt.xlim(0, 1)
plt.ylim(0, 0.51)
plt.xticks(np.arange(min(d1_alpha_rho[0, :]), max(d1_alpha_rho[0, :])+0.1,
    0.1))
plt.xlabel(r'$\alpha$', fontsize=20)
plt.ylabel(r'$\rho$', fontsize=20)

# add the colorbar

colorbar=plt.colorbar()
colorbar.ax.get_yaxis().labelpad = 20
colorbar.ax.set_ylabel('$C$', rotation=90)

plt.savefig('facebook_data_colorbargraph.eps')
plt.show()
```

A.10 Plotting Figure 4.3, α versus ρ for Facebook data with initial sampling and varied λ

```
#import all the required packages

import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
from time import time
import matplotlib as mpl

# set the standard parameters

params = { 'figure.figsize': (10, 6),
     'axes.labelsize': 20,
          'text.fontsize': 24,
               'xtick.labelsize': 18,
                    'ytick.labelsize': 18,
                         'legend.fontsize': 18,
                              'text.usetex': False,
```

```
                                 'mathtext.bf': 'helvetica:bold',
                        }


plt.rcParams.update(params)


# create a colorbar

col01=[165./255., 0./255., 38./255.]
col1=[215./255., 48./255., 39./255.]
col2=[244./255., 109./255., 67./255.]
col3=[253./255., 174./255., 97./255.]
col4=[254./255., 224./255., 144./255.]
col41=[255./255., 255./255., 191./255.]
col42=[224./255., 243./255., 248./255.]
col51=[171./255., 217./255., 233./255.]
col52=[116./255., 173./255., 209./255.]
col6=[69./255., 117./255., 180./255.]
col61=[49./255., 54./255., 149./255.]

col0=np.array([col01, col1, col2, col3, col4, col41, col42, col51, col52, col6, col61])

cm = mpl.colors.ListedColormap(col0)

# load the necessary data

d1_alpha_rho1=np.loadtxt('facebook_subgraph_divide10.txt')
d1_alpha_rho2=np.loadtxt('facebook_subgraph_divide20.txt')
d1_alpha_rho3=np.loadtxt('facebook_subgraph_divide30.txt')
d1_alpha_rho4=np.loadtxt('facebook_subgraph_divide40.txt')

# create a subplot for different values of lambda

f, axes = plt.subplots(2, 2)

# format the plot

super_axis = f.add_subplot(111)
super_axis.set_axis_bgcolor('none')
super_axis.axes.get_xaxis().set_ticks([])
super_axis.axes.get_yaxis().set_ticks([])
super_axis.tick_params(labelcolor='none', top='off', bottom='off', left='off', right='off')
super_axis.spines['bottom'].set_color('none')
super_axis.spines['top'].set_color('none')
super_axis.spines['left'].set_color('none')
super_axis.spines['right'].set_color('none')

# lambda= 1/10

axes[0, 0].scatter(d1_alpha_rho1[0, :], d1_alpha_rho1[1, :], s=100, c=d1_alpha_rho1[3, :], cmap=cm, marker='o', vmin=0, vmax=0.55)
axes[0, 0].axhline(y=.25, xmin=0.0, xmax=1.0, ls='--', color='grey')
```

```python
axes[0, 0].text(0.1, 0.3, 'A', transform=axes[0, 0].transAxes, fontsize=24, fontweight='bold', va='top', ha='right')
axes[0, 0].set_ylabel(r'$\rho$')
axes[0, 0].annotate(r'$\lambda=\frac{1}{10}$', xy=(1, 0.01), xycoords='axes fraction', fontsize=15, xytext=(-5, 5), textcoords='offset points', ha='right', va='bottom')


# lambda=1/20

axes[0, 1].scatter(d1_alpha_rho2[0, :], d1_alpha_rho2[1, :], s=100, c=d1_alpha_rho2[3, :], cmap=cm, marker='o', vmin=0, vmax=0.55)
axes[0, 1].axhline(y=.25, xmin=0.0, xmax=1.0, ls='--', color='grey')
axes[0, 1].text(0.1, 0.3, 'B', transform=axes[0, 1].transAxes, fontsize=24, fontweight='bold', va='top', ha='right')
axes[0, 1].annotate(r'$\lambda=\frac{1}{20}$', xy=(1, 0.01), xycoords='axes fraction', fontsize=15, xytext=(-5, 5), textcoords='offset points', ha='right', va='bottom')

#lambda=1/30

axes[1, 0].set_ylabel(r'$\rho$')
axes[1, 0].scatter(d1_alpha_rho3[0, :], d1_alpha_rho3[1, :], s=100, c=d1_alpha_rho3[3, :], cmap=cm, marker='o', vmin=0, vmax=0.55)
axes[1, 0].axhline(y=.25, xmin=0.0, xmax=1.0, ls='--', color='grey')
axes[1, 0].text(0.1, 0.3, 'C', transform=axes[1, 0].transAxes, fontsize=24, fontweight='bold', va='top', ha='right')
axes[1, 0].set_xlabel(r'$\alpha$')
axes[1, 0].annotate(r'$\lambda=\frac{1}{30}$', xy=(1, 0.01), xycoords='axes fraction', fontsize=15, xytext=(-5, 5), textcoords='offset points', ha='right', va='bottom')

# lambda=1/40

p1=axes[1, 1].scatter(d1_alpha_rho4[0, :], d1_alpha_rho4[1, :], s=100, c=d1_alpha_rho4[3, :], cmap=cm, marker='o', vmin=0, vmax=0.55)
axes[1, 1].axhline(y=.25, xmin=0.0, xmax=1.0, ls='--', color='grey')
axes[1, 1].text(0.1, 0.3, 'D', transform=axes[1, 1].transAxes, fontsize=24, fontweight='bold', va='top', ha='right')
axes[1, 1].set_xlabel(r'$\alpha$')
axes[1, 1].annotate(r'$\lambda=\frac{1}{40}$', xy=(1, 0.01), xycoords='axes fraction', fontsize=15, xytext=(-5, 5), textcoords='offset points', ha='right', va='bottom')

# format the axes

axes[0, 0].set_position([0.1, 0.5, 0.35, 0.35])
axes[0, 1].set_position([0.525, 0.5, 0.35, 0.35])
axes[1, 0].set_position([0.1, 0.1, 0.35, 0.35])
axes[1, 1].set_position([0.525, 0.1, 0.35, 0.35])

plt.sca(axes[0, 0])
plt.xticks([0.3, 0.6, 0.9])
plt.sca(axes[0, 0])
plt.yticks([0.0, 0.25, 0.5])
```

```
plt.ylim(-0.01, 0.52)

plt.sca(axes[0, 1])
plt.xticks([0.3, 0.6, 0.9])
plt.sca(axes[0, 1])
plt.yticks([0.0, 0.25, 0.5])
plt.ylim(-0.01, 0.52)

plt.sca(axes[1, 0])
plt.xticks([0.3, 0.6, 0.9])
plt.sca(axes[1, 0])
plt.yticks([0.0, 0.25, 0.5])
plt.ylim(-0.01, 0.52)

plt.sca(axes[1, 1])
plt.xticks([0.3, 0.6, 0.9])
plt.sca(axes[1, 1])
plt.yticks([0.0, 0.25, 0.5])
plt.ylim(-0.01, 0.52)

# add the colorbar

cbar_ax = f.add_axes([0.9, 0.285, 0.028, 0.4])
cbar=f.colorbar(p1, cax=cbar_ax,ticks=([0.05, 0.25, 0.45]))
cbar.ax.tick_params(labelsize=10)
cbar.set_label(r'$C$', fontsize=20)

plt.savefig('facebook_diff_lambda.eps')
plt.show()
```

Bibliography

[1] Gesa A. Böhme and Thilo Gross. Fragmentation transitions in multistate voter models. *Phys. Rev. E*, 85:066117, Jun 2012.

[2] Damon Centola. The spread of behavior in an online social network experiment. *Science*, 329(5996):1194–1197, 2010.

[3] Iain D. Couzin, Christos C. Ioannou, Güven Demirel, Thilo Gross, Colin J. Torney, Andrew Hartnett, Larissa Conradt, Simon A. Levin, and Naomi E. Leonard. Uninformed individuals promote democratic consensus in animal groups. *Science*, 334(6062):1578–1580, 2011.

[4] Richard Durrett, James P. Gleeson, Alun L. Lloyd, Peter J. Mucha, Feng Shi, David Sivakoff, Joshua E. S. Socolar, and Chris Varghese. Graph fission in an evolving voter model. *Proceedings of the National Academy of Sciences*, 109(10):3682–3687, 2012.

[5] Rick Durrett. *Random Graph Dynamics*. Cambridge University Press, 2007.

[6] Juan Fernández-Gracia, Krzysztof Suchecki, José J. Ramasco, Maxi San Miguel, and Víctor M. Eguíluz. Is the voter model a model for voters? *Phys. Rev. Lett.*, 112:158701, Apr 2014.

[7] Y. Gandica, A. Charmell, J. Villegas-Febres, and I. Bonalde. Cluster-size entropy in the axelrod model of social influence: Small-world networks and mass media. *Phys. Rev. E*, 84:046109, Oct 2011.

[8] Thilo Gross, Carlos J. Dommar D'Lima, and Bernd Blasius. Epidemic dynamics on an adaptive network. *Phys. Rev. Lett.*, 96:208701, May 2006.

[9] Beniamino Guerra, Julia Poncela, Jesús Gómez-Gardeñes, Vito Latora, and Yamir Moreno. Dynamical organization towards consensus in the axelrod model on complex networks. *Phys. Rev. E*, 81:056105, May 2010.

[10] Petter Holme and M. E. J. Newman. Nonequilibrium phase transition in the coevolution of networks and opinions. *Phys. Rev. E*, 74:056108, Nov 2006.

[11] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, Jun, 2014.

[12] Kevin T. Macon, Peter J. Mucha, and Mason A. Porter. Community structure in the united nations general assembly. *Physica A: Statistical Mechanics and its Applications*, 391(1–2):343 – 361, 2012.

[13] Nishant Malik and Peter J. Mucha. Role of social environment and social clustering in spread of opinions in coevolving networks. *Chaos*, 23(4), 2013.

[14] Roberto A. Monetti. First-order irreversible phase transitions in a nonequilibrium system: Mean-field analysis and simulation results. *Phys. Rev. E*, 65:016103, Dec 2001.

[15] Mehdi Moussaïd, Juliane E. Kämmer, Pantelis P. Analytis, and Hansjörg Neth. Social in-

fluence and the collective dynamics of opinion formation. *PLoS ONE*, 8(11):1–8, November 2013.

[16] Peter J. Mucha, Thomas Richardson, Kevin Macon, Mason A. Porter, and Jukka-Pekka Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *Science*, 328(5980):876–878, 2010.

[17] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

[18] Gergely Palla, Albert-Laszlo Barabasi, and Tamas Vicsek. Quantifying social group evolution. *Nature*, 446(7136):664–667, 04 2007.

[19] Politico.com. 2016 presidential primaries results. `http://www.politico.com/2016-election/results/map/president`, May 2016.

[20] C. E. La Rocca, L. A. Braunstein, and F. Vazquez. The influence of persuasion in opinion formation and polarization. *EPL (Europhysics Letters)*, 106(4):40004, 2014.

[21] Feng Shi, Peter J. Mucha, and Richard Durrett. Multiopinion coevolving voter model with infinitely many phase transitions. *Phys. Rev. E*, 88:062818, Dec 2013.

[22] Gábor Szabó, Mikko Alava, and János Kertész. Structural transitions in scale-free networks. *Phys. Rev. E*, 67:056102, May 2003.

[23] James A. Thomson and Jesse Sussell. Is geographic clustering driving political polarization? *The Washington Post*, March 2, 2015.

[24] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 06 1998.

[25] Damián H. Zanette and Susanna C. Manrubia. Vertical transmission of culture and the distribution of family names. *Physica A: Statistical Mechanics and its Applications*, 295(1–2):1 – 8, 2001. Proceedings of the {IUPAP} International Conference on New Trends in the Fractal Aspects of Complex Systems.

[26] Gerd Zschaler, Gesa A. Böhme, Michael Seißinger, Cristián Huepe, and Thilo Gross. Early fragmentation in the adaptive voter model on directed networks. *Phys. Rev. E*, 85:046107, Apr 2012.