Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Lecture with Computer Exercises:
Modelling and Simulating Social Systems with MATLAB

Project Report

Network Based Modelling for the Spread of Scientific Ideas

Mayra Bermúdez, Sarah Grimm & Sayed-Rzgar Hosseini

Zurich
December 14th, 2012

Agreement for free-download

We hereby agree to make our source code for this project freely available for download from the web pages of the SOMS chair. Furthermore, we assure that all source code is written by ourselves and is not violating any copyright restrictions.

Mayra Bermúdez Sarah Grimm Sayed-Rzgar Hosseini

Contents

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of Originality

This sheet must be signed and enclosed with every piece of written work submitted at ETH.

I hereby declare that the written work I have submitted entitled

Network-Based Modelling for the Spread of Scientific Ideas

is original work which I alone have authored and which is written in my own words.*

Author(s)

Last name	First name
Bermúdez	Mayra
Grimm	Sarah
Hosseini	Sayed-Rzgar

Supervising lecturer

Last name	First name

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (http://www.ethz.ch/students/exams/plagiarism_s_en.pdf). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Dec. 13, 2012
Place and date

[illegible]
[illegible]
[illegible]
Signature

*Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Print form

1 Abstract

Using simulations, we investigated the changes in networks with nodes that held different ‘scientific ideas’ and influenced each other through a complex contagion mechanism. This was divided into two parts: by varying the network structure and observing features of the distribution of ideas in networks, and by varying the starting idea distributions of the nodes and observing features of the network structures. The update step included a rewiring probability, a complex contagion threshold, and a probability of innovation (producing a new idea). We found that both structures and idea distributions influenced each other’s features. Values of the update parameter also played a role.

2 Individual contributions

This report represents a group effort by all members.

3 Introduction and Motivation

We live in a time in which aspects of our lives and of our world become more and more connected with each other. To understand one aspect, we must understand many other aspects, which all together form a large, complex system, or network. Globalization has changed the meaning of 'distance' and communication, allowing seemingly unrelated and unconnected individuals to share more than they ever could before. Fields of studies are overlapping with each other, creating new interdisciplinary domains and building a diverse playground for the sharing of ideas. But how do ideas spread? This question is especially interesting with the increase of technology that allows us to record and visualize the networks that connect individuals and ideas, and in particular the ability to 'see' how they change. This can lead to insight about why and when ideas spread and into the complexity of the matter. Research has shown that not only does the nature of information or innovation influence the diffusion of it, but also that the structure of a network influences the diffusion dynamics. Here, we try to simulate the spread of scientific ideas in different networks. The model presented is based on two studies: one that investigated critical parameter values for complex contagion (Centola and Macy, 2007) and another that investigated critical values of a rewiring parameter (Holme and Newman, 2006).

3.1 Fundamental Questions

The main goals of this simulation study are to investigate how network structure influences the distribution of ideas, and how the distribution of ideas influences network structure. For a list of the terminology that will be used throughout the paper, please refer to Table 1. By varying three parameters - probability of rewiring, rate of innovation, and complex contagion threshold (ϕ, α, and δ respectively) - and using different network structures and idea distributions, we observed how network structure characteristics changed. Similarly, we observed how the distribution of ideas and the connections between them changed. Below we describe our questions more specifically.

Table 1: List of terminology

	Term
Neighborhood index	The fraction of the holders of the same idea who are neighbor as well averaged over all ideas.
Intra-idea distance	The average distance between holders of the same idea in the network.
Dominant frequency	The frequency of the dominant idea in the network at each time steps of the simulation.
Average dominance time	The average number of time steps in which the dominant idea keeps its dominance.
Novelty index	The fraction of newly generated ideas.
Average shortest path	The average number of steps along the shortest paths for all possible pairs of network nodes.
Clustering coefficient	A measure of degree to which nodes in a graph tend to cluster together.
Degree of connectivity	The number of edges incident to the vertex.
Connected component	A subgraph in which any two vertices are connected to each other by paths, and which is connected to no additional vertices in the supergraph.
Diameter of network	The longest of all the calculated shortest paths in a network.

3.1.1 Effects of Network Structure on Idea Distribution

Given a starting network and a random idea distribution, how do different network structures (see Section **??**) affect the distance between nodes that have the same idea (intra-idea distance)? How do they affect the neighbourhood index? How do they change the emergence of dominant ideas and their time of dominance? How do their effects depend on the values of ϕ, α and δ?

More rigid network structures (those with less ‘randomness’, such as the caveman and the small world networks) may make it more difficult for ‘like-minded’ nodes (that is, nodes with the same idea) to connect and may thus have smaller neighbourhood indexes and larger intra-idea distances than the more random network structures (such as the random and scale-free networks). Their effects may be more sensitive to the values of ϕ (because this affects how likely it is for their structure to change) and to values of δ because being restricted to a more closed group of nodes makes it difficult to reach a threshold necessary to become similar to surrounding nodes. Values of α may decrease the neighbourhood indexes by creating larger diversity among neighbouring nodes.

If more rigid network structures do make it more difficult for like-minded nodes to connect, then it would be more difficult for a dominant idea to emerge in these networks. These effects may be smaller for larger values of ϕ since these values would allow for the structure to change more. For larger values of ϕ therefore one could expect that the effects of the network structures on the characteristics of the idea distribution are more similar since allowing to change the structure removes their initial influence.

3.1.2 Effects of Idea Distribution on Network Structure

Given a starting idea distribution and a caveman network structure, how do different idea distributions affect the average path length and diameter of the network? Do they change the number of connected components in the network? Do clusters form differently, and how does the clustering coefficient change? What does the distribution of node degree looks like? How do these effects depend on the values of ϕ, α, and δ?

If the starting idea distribution is parallel to the caveman network structure (see Section 4.1.1 for a description of idea distributions), then like-minded nodes will already be connected and thus rewiring will probably not change much of the average path length, nor will it change the network diameter. Similarly, the clustering coefficient will remain high just like the starting value. The distribution of the node degree will also not change (nodes will have one of two values for their degree). In other words, if the idea distribution is parallel to the network structure, the structure will not change much. Changing ϕ and δ will not change these effects, and perhaps increasing α will decrease the clustering coefficient and will increase the number of connected components because nodes will disconnect from nodes with novel ideas and will rewire to nodes with the same idea.

If the starting idea distribution is random, then the network's caves will disintegrate as nodes will rewire with other nodes outside of their caves. This will change the degree distribution by increasing its variance (because nodes will have a variety of different degree values). Depending on the value of δ this disintegration may be reduced because nodes have a higher chance of forming dominant ideas within caves. Similarly, increasing ϕ will increase the disintegration of caves. Thus, for this idea distribution the parameter values may play a larger role.

If the starting idea distribution is anti-parallel, nodes within each cave will initially be connected with nodes that do not hold the same idea as them. Therefore the threshold δ will not be met in order for nodes to change their ideas, and they will rewire with other nodes outside of their cave. The clustering coefficient, as well as the number of connected components, the network diameter, and the average path length, will likely decrease since the structure will change significantly. The degree distribution will increase in variance. Increasing ϕ and α and decreasing δ will probably increase the magnitude of these effects. Thus, having an anti-parallel idea distribution will probably display the most changes in the characterstics of the network structure that are in question.

4 Description of the Model

The model used here is based on a study by Holme and Newman (2006). Each simulation begins with a specified network structure as well as a distribution of the 'idea' (or state) of the nodes. At each time step a node either changes its idea to that of one of its neighbours' ideas if its frequency surpasses a defined threshold, rewires to connect with a node that has the same idea, or generates a novel idea (this is the innovation parameter).

Given the network structure and node states, three parameters are introduced: ϕ (probability of rewiring), α (probability of innovation), and δ (complex contagion threshold). As in Holme and Newman (2006), ϕ is a value from zero to one, and is the probability that one of the edges of a randomly chosen node i will be changed to connect to another node j that i is unconnected with. We decided to add one more criterion to this definition: node j is a node that has the same idea as node i. This encourages the simulations to reflect a common tendency of individuals to seek out others who think like them. If there is no such node j, then the chosen node will do nothing.

At each time step a node may 'come up with a new idea' with a probability of α. This value is small to reflect that novel ideas are not frequently observed.

We introduced a node threshold δ to the general model in order to investigate the behaviour of complex contagion as opposed to simple contagion. This was motivated by a study by Centola and Macy (2007). Simple contagion is well suited for modeling the spread of diseases since they may often be passed on by a single contact with an infected individual. However, as our intuition may suggest, other kinds of innovations raise questions about the legitimacy and credibility of the innovations themselves, and may thus require exposure to multiple sources of the innovation. This is called complex contagion. Models usually represent it in two ways regarding the number of connected sources that must have adopted an innovation in order to influence the agent (or individual) in question: it is either a fixed number (greater than one) or a fraction (between zero and one, inclusive). Our simulation used the second formulation since the degree of individual nodes varied across network structures.

4.1 Networks

For the purposes of our simulations, we used four network structures. These structures can be characterized by properties such as average shortest path lengths, clustering coefficients, and the degree of connectivity (see Table 1 for definitions). Below are short descriptions of each network structure. In order to compare between different structures, the network structure parameters were chosen such that the mean degree of the networks were similar (approximately 25). For further details about parameter values, see Table 2.

Table 2: Table of parameters used in simulations.

Network	Parameter	Value		
	ϕ	0.1	0.3	0.5
	α	0.01	0.05	0.1
	δ	0.001	0.01	0.05
	n	1000		
	t_{end}	1000		
Caveman	m	40		
	p	40		
Random (Erdos-Renyi)	$prob$	0.025		
Scale free	m_0	24		
	m_1	12		
Small world	κ	24		
	β	0.1		

4.1.1 Random Graph

Random graphs have a short average path length. The graph is defined by the total number of nodes, and by the probability of any two nodes to be connected. Thus, all connections are random. They typically have a small clustering coefficient. Here we used a variant of the Erdős-Rényi random graph model (Erdős and Rényi, 1960) as implemented by Brugger and Schwirzer (2011).

4.1.2 Caveman Graph

The caveman structure, as defined by Watts (2003), has k isolated and fully connected 'cliques' from which one link is changed to connect one clique to another, rendering all cliques to be connected. Thus, relative to random graphs, they have a high clustering coefficient and a large average shortest path length.

4.1.3 Small World Graph

Small world graphs have characteristics that lie in between random graphs and highly clustered graphs (such as caveman graphs): they have a high clustering coefficient similar to the latter, but also have a small average shortest path similar to the former. Many real-world networks have been observed to have a small world structure, and thus we included it in our simulations. Here we used the graph as defined by Watts and Strogatz (1998), and implemented by Brugger and Schwirzer (2011).

4.1.4 Scale-Free Graph

Scale-free network structures are often found where new nodes are constantly being added, and they are connected to already well-connected nodes. Such a structure displays a scale-free power-law distribution of the degree (connectivity) of nodes. Thus, there are few nodes that are highly connected, and more nodes that are moderately or mildly connected. Compared to random graphs, they have a smaller average shortest path. This graph was implemented by Brugger and Schwirzer (2011) as defined by Barabási and Albert (1999).

4.2 Ideas

After choosing a starting structure for our model, we then chose a distribution for the starting ideas (states) of nodes. Each node was randomly assigned one of these ideas, thus allowing for multiple nodes to have the same idea. For the caveman structure, however, there were two other options: to either distribute the starting ideas 'parallel' to the structure, i.e. such that all nodes in a cave shared the same idea, or 'anti-parallel' such that all nodes in a cave had a different idea. This was used for the analysis of the effect of the idea distribution on the network structure. Why was the caveman structure investigated? There were two reasons: firstly, it was straightforward how to define idea distributions that are in accord or disaccord with the caves in the network. Secondly, in the scientific community, research teams may often be made up of closely-connected members that are only weakly connected to other research teams, and within these teams, members may or may not be interested in the same ideas for research.

As previously mentioned, each node had a small probability α of adopting a novel idea from a virtually unlimited number of new ideas.

5 Implementation

Our simulation comprises mainly two parts: the phase 1 corresponding to the study of the effect of network structure on the idea distribution, and phase 2 where we study how the distribution of the ideas affect the topology of the network. These two phases are implemented in the same MATLAB file **mainscript.m**. Implementation of both phases is divided into three steps.

Step 1: Network structure is chosen and we generate the adjacency matrix corresponding to that structure. The initial idea distribution is also generated.
Step 2: The updating process is done onto the chosen network structure.
Step 3: At this step a series of functions are called to get the results.

5.1 Step 1: Generating network structures and initial idea distribution

For the first phase, one of the functions **step1_scalefree**, **step1_caveman**, **step1_smallworld** and **step1_randomgraph** is called, depending on our choice. Each function is found in the MATLAB files **step1_scalefree.m, step1_caveman.m, step1_randomgraph.m** and **step1_smallworld.m** correspondingly. Each of these functions generates the adjacency matrix corresponding to the chosen network structure. Also in this step, the initial idea distribution - a random distribution for the first phase - is applied onto the nodes.
For the second phase we generated just the adjacency matrix for the caveman structure network, and then we chose among three different initial idea distributions: random, parallel or anti-parallel. After choosing one of them, the initial idea distribution vector is generated.

5.2 Step 2: Update

This step is the same for both phases.

The updating rules in the simulation are implemented in the file **step2.m** and done by function **step2** which requires the following parameters:

- `t_end`: number of iterations
- `phi`: network reorganization rate
- `alpha`: innovation rate
- `mat`: initial connectivity matrix
- `vec`: initial idea vector

- `p`: initial number of opinions

And has as outputs:

- `mat`: connectivity matrix after simulation
- `vec`: idea vector after simulation
- `dominant_freq`: the vector holding the frequency of the dominant idea
- `most_freq`: the vector holding the index of the dominating idea in each time step

The function **step2**, in which the updating process is executed, implements the following algorithm:

Algorithm 1 Update process

```
for each of the iterations do
    choose a node $x1$ in the network at random
    generate a random number $a1$
    if $a1 < phi$ then                    ▷ With probability phi we reorganize the network
        eliminate the connection between $x1$ and one of its neighbours with a different idea
        select another node at random among the nodes with the same idea as $x1$ and that are not already neighbours of $x1$ and create a connection between them
    else change the idea of $x1$ to one of the ideas of its neighbours which meet the threshold
    end if
    choose a node $y$ in the random to come up with a novel idea
    generate a random number $a2$
    if $a2 < \alpha$ then $y$ comes up with a new idea    ▷ With probability alpha y comes up with a new idea
    end if
end for
```

5.3 Step 3: Getting results

In the phase 1 we want to observe the influence of a certain network structure on the distribution of ideas after the updating process (step 2). For this, it is necessary to search for features that reflect the final distribution of ideas. In this study we observed the following features:

- `n_index`: the average neighbor index of the network
- `intra_idea_distance`: the average of the average shortest distance between agents holding the same idea
- `average_dominance_time`: the average of the dominance time for different dominance periods

All these parameters are obtained in **step3a.m, step3b.m** and **step3d.m** which are called in **main_script.m**.

In phase 2 we want to find out how the distribution of ideas changes the network structure. For this reason we look at a fixed network structure: the caveman structure. After step 2 we want to see how the initial structure was modified by the updating process, and in order to do this we observed the following features:

- `clust_coeff`: the clustering coefficient of the network
- `[dgr,frq]`: degree vector and its corresponding frequency vector
- `average_path_length`: the average path length for the graph
- `graph_diameter`: outputs the diameter of the graph

6 Simulation Results and Discussion

For the first section of the results, ideas were assigned randomly onto the nodes of four different network structures (caveman, small world, random, and scale-free). Chapter 4 describes the network structures and idea distributions that are discussed. Simulations were run for each network structure for 27 different parameter combinations (three values for each of ϕ, δ and α). The effects of each network structure on the idea distribution was evaluated on five features: the average dominance time of dominant (i.e., most frequent) ideas, the novelty index, the intra-idea distance, the neighbourhood index, and the frequency of dominance of an idea (see Table 1). We also investigated how these effects varied with the three parameters (ϕ, δ, α).

The second section of the results investigated the opposite direction of effects: how changing the idea distribution affected the resulting network structure. To do this we applied three different idea distributions (random, parallel, and anti-parallel to the structure) to a caveman network (see Table 1) with 40 caves. There were again 27 different parameter combinations for each idea distribution, as in the previous analysis. The five features of the network structure that we evaluated were: the clustering coefficient, the degree distribution of the nodes, the number of connected components, the average path length, and the network diameter.

We found that there were different behaviours for some features of the idea distribution depending on the network structure, and that there were different behaviours for the structure features depending on the idea distribution. Both of these results were at least partly influenced by the values of ϕ, δ, and α.

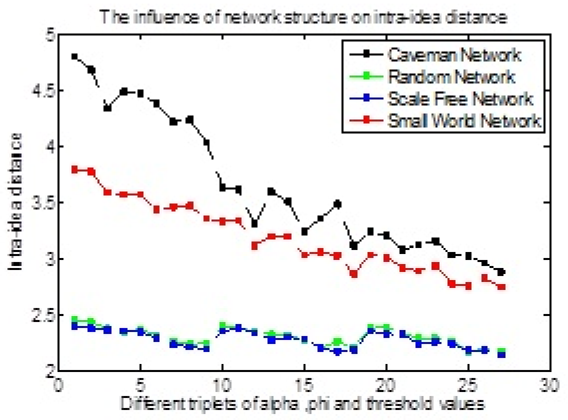
6.1 Effects of Network Structure on Idea Distribution

The results of the average dominance time and the novelty index were too dependent on the parameter values and were not included; the network structure does not seem to play a strong enough role over all parameter combinations in influencing these two features. It is not surprising, however, that the novelty index was highly dependent on the value of α, but it is not so intuitive why the average dominance time was not correlated with α at all: increasing the number of novel ideas decreases the possible number of 'followers' for already-established ideas, which should affect the frequency of dominance and therefore the dominance time. Perhaps the value of α would need to be increased to observe this behaviour.

Furthermore, we observed two different results for the intra-idea distance and the neighbourhood index: those from the caveman and small world structures, and those from the random and scale-free structures. All three parameter values affected these results.

6.1.1 Intra-Idea Distance and Neighbourhood Index

For any parameter combination, the caveman and small world structures resulted in larger intra-idea distances (respectively) than those of the random and scale-free structures, which were very similar to each other (Figure 1). Interestingly, a similar influence was found on the neighbourhood index: the caveman structure held the largest index regardless of parameter combination, followed by the small world, random, and scale-free structures (Figure 1). It seems that the caveman structure encourages nodes to be within the direct neighbourhood of like-minded nodes (nodes with the same idea) and at a farther distance from like-minded nodes that are not in their direct neighbourhood, whereas the random and scale-free structures have a tendency to keep like-minded nodes in each other's direct neighbourhood but to also keep those like-minded nodes not in their direct neighbourhood at a shorter distance. The difference in intra-idea distance between network structures could be a result of the general smaller average path distance that random and scale-free networks have as compared to the caveman and small world graphs.



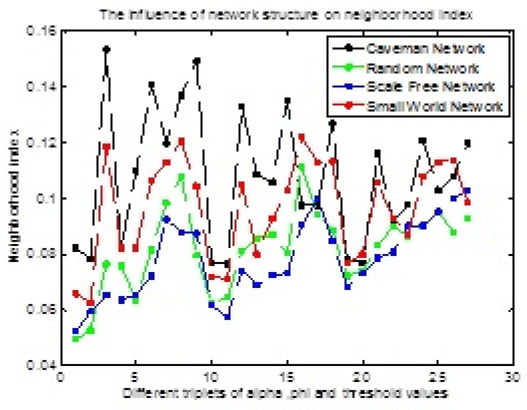


Figure 1: The influence of network structure on intra idea distance (left) and neighborhood index (right) for 27 distinct sets of parameters (triplets of alpha-phi-threshold).

Dependence on α For all network structures, increasing the level of innovation α decreased the intra-idea distance. This effect was more pronounced in the scale-free and random networks (Figure 2). Does innovation bring people together in scientific communities? Perhaps, and perhaps not. It is possible that these results are due to the construction of our model: because the mechanism of influence involves complex contagion, new ideas would not be able to spread (since they are held only by the originator) and therefore these ideas would technically have an intra-idea distance of zero.

Higher values of α also increased the neighbourhood index for all network structures. This correlation was again most visible for the scale-free network, followed by the random network (Figure 3). One possible explanation for this correlation is that the more novel ideas nodes create, the less likely it is that the contagion threshold is met for other ideas, and thus nodes will only be rewiring instead of also changing their ideas. Thus more like-minded nodes will be connected, and the index increases.

Dependence on ϕ By increasing ϕ, the intra-idea distance decreased for all network structures. This correlation is quite an intuitive result since ϕ is the probability of deleting a connection between two nodes with different ideas and the formation of a new connection between two nodes with the same idea, and thus, by increasing ϕ the distance between like-minded nodes decreases. Unlike the α parameter, the effects of ϕ are more pronounced for the caveman and small world structures rather than for the random and scale-free networks (Figure 4).

On the other hand, the effects of ϕ on the neighborhood index varied between networks. Increasing ϕ increased the neighbourhood indexes of the random and scale-free networks, while it decreased the neighbourhood index of the caveman network and had no correlation with changes in the small world network (Figure 5).

Dependence on δ Increasing values of the complex contagion threshold δ slightly decreased the intra-idea distance for the caveman and small world network structures (Figure 6).

No correlation was found between the neighbourhood index of the small world, scale-free, and random network structures and values of δ. However, the neighbourhood index for the caveman network increased as δ increased (Figure 7). This is an interesting result. On one hand, it is intuitive since requiring more like-minded nodes to be in the direct neighbourhood for a node to adopt their idea automatically increases the number of like-minded nodes in the neighbourhood (if the node adopts their idea). However, increasing δ could have the opposite effect: if the threshold is too high, nodes will not adopt their neighbhours' ideas and thus the neighbourhood index would remain small. The influence of δ only on the caveman network could be due to this structure's higher degree per node: almost all nodes have the same degree, whereas the other structures have the same average amount but with greater variance.

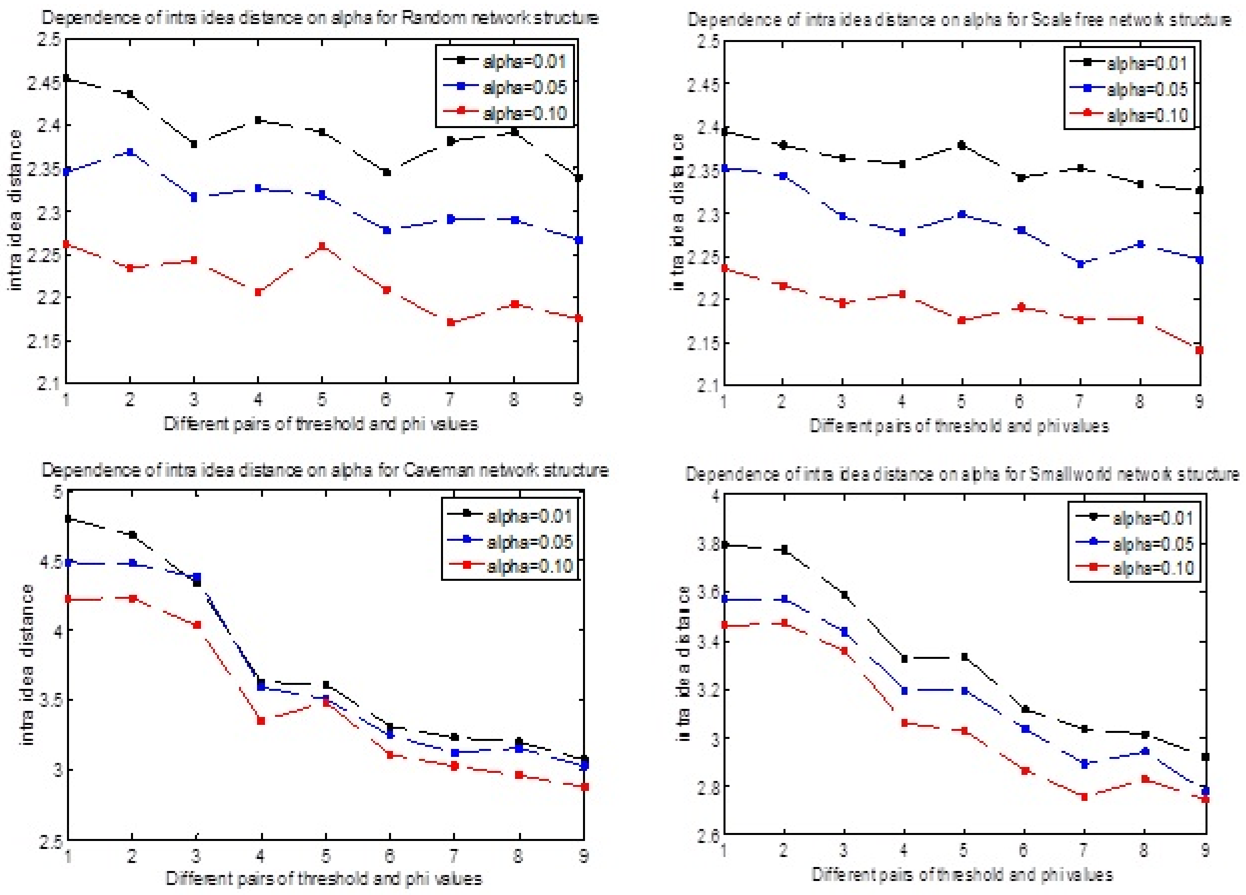


Figure 2: Dependence of intra idea distance on alpha for each network structure. Horizontal axis corresponds to distinct pairs of phi-threshold and vertical axis is the intra idea distance. Each curve corresponds to a different value of alpha. Black curve corresponds to the smallest value of alpha and it always has the highest intra idea distance while the red curve corresponds to the largest value of alpha which always has the lowest value of intra idea distance.

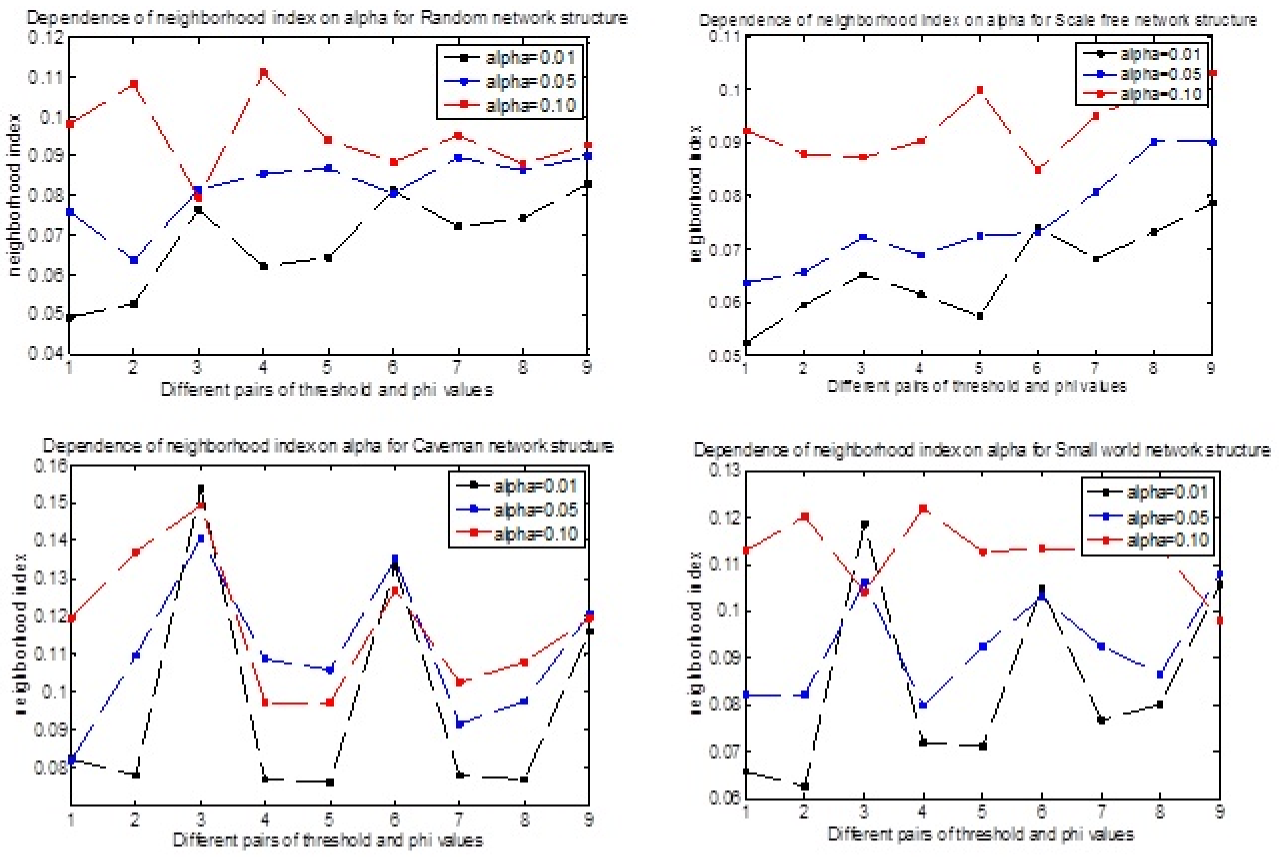


Figure 3: Dependence of neighborhood index on alpha for each network structure. Horizontal axis corresponds to distinct pairs of phi-threshold and vertical axis is the neighborhood index. Each curve corresponds to a different value of alpha. Red curve corresponds to the largest value of alpha and it has the highest neighborhood index while the black curve corresponds to the smallest value of alpha which has the lowest value of intra idea distance.

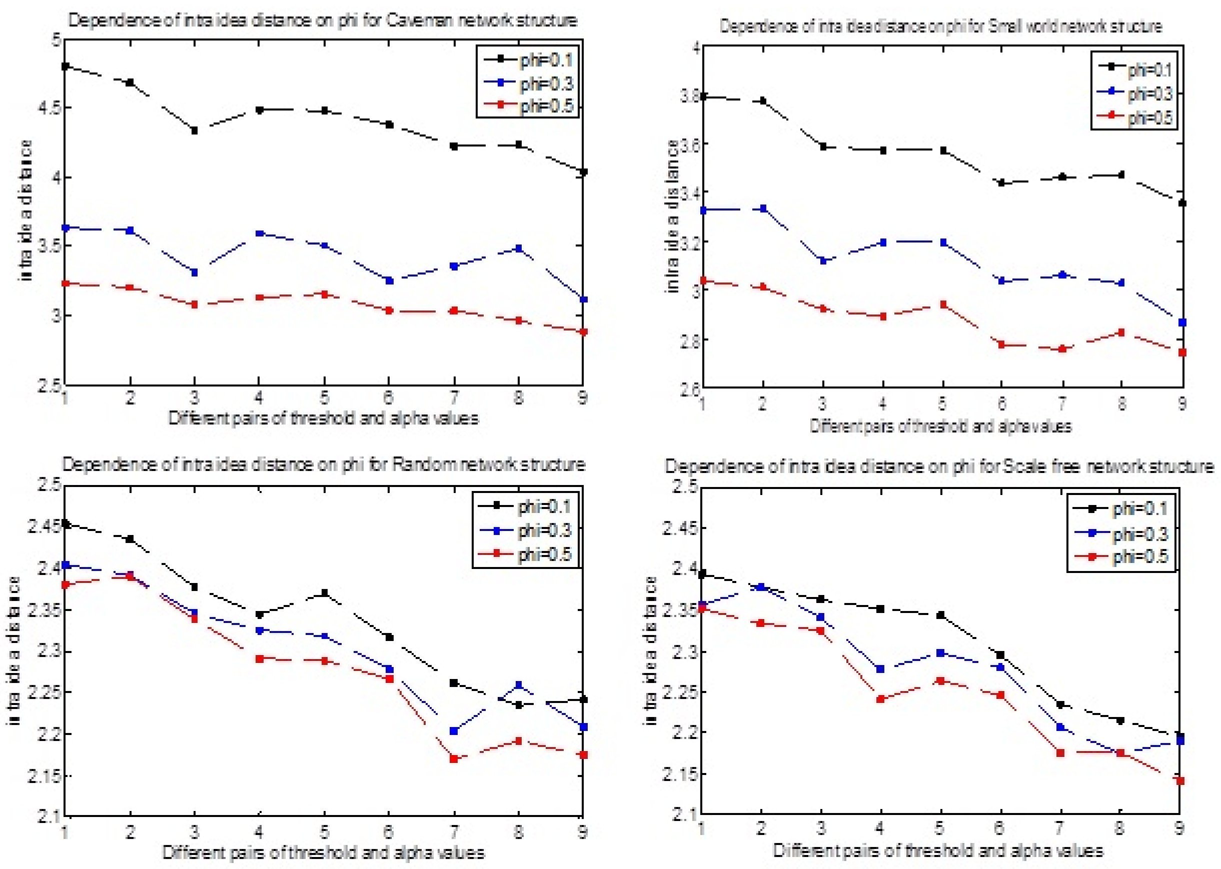


Figure 4: Dependence of intra idea distance on phi for each network structure. Horizontal axis corresponds to distinct pairs of alpha-threshold and vertical axis is the intra idea distance. Each curve corresponds to a different value of phi. Black curve corresponds to the smallest value of phi and it always has the highest intra idea distance while the red curve corresponds to the largest value of phi which always has the lowest value of intra idea distance.

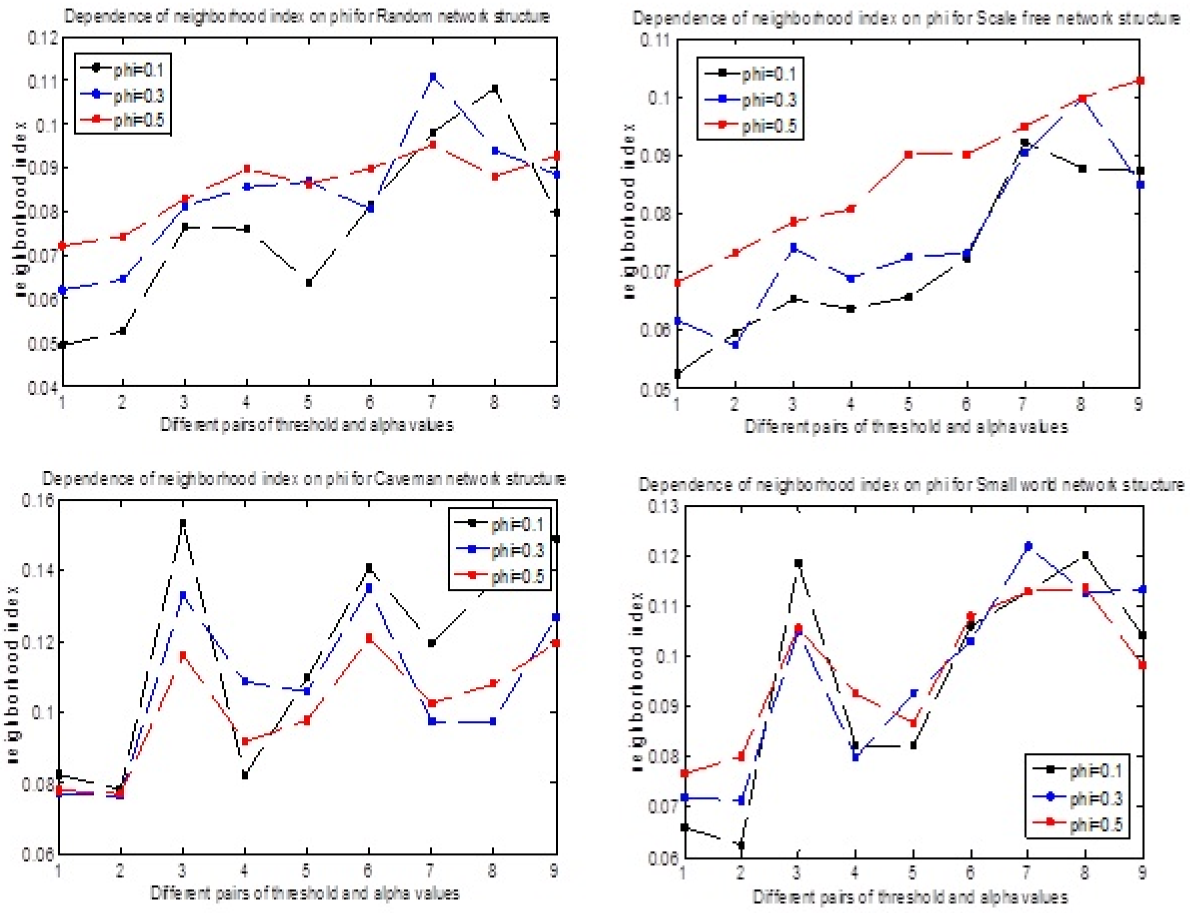


Figure 5: Dependence of neighborhood index on phi for each network structure. Horizontal axis corresponds to distinct pairs of alpha-threshold and vertical axis is the neighborhood index. Each curve corresponds to a different value of phi. Increasing phi leads to increase neighborhood distance for Random and Scale free networks while increasing phi causes the neighborhood index to be decreased for Cavemen Network. For Small world network there is no significant dependence of neighborhood index on phi.

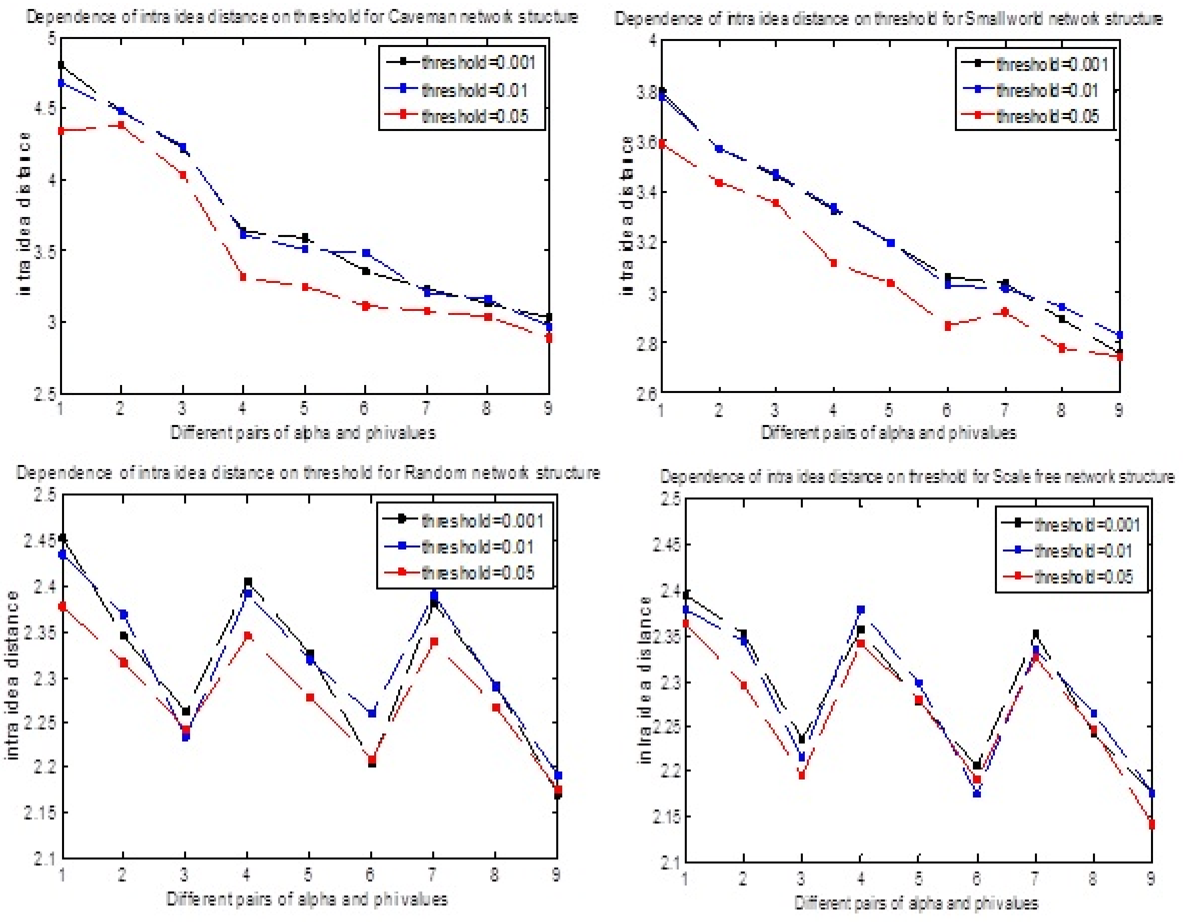


Figure 6: Dependence of intra idea distance on complex contagion threshold for each network structure. Horizontal axis corresponds to distinct pairs of alpha-phi and vertical axis is the intra idea distance. Each curve corresponds to a different value of complex contagion threshold. Black curve corresponds to the smallest value of complex contagion threshold and it has the highest intra idea distance while the red curve corresponds to the largest value of complex contagion threshold which has the lowest value of intra idea distance.

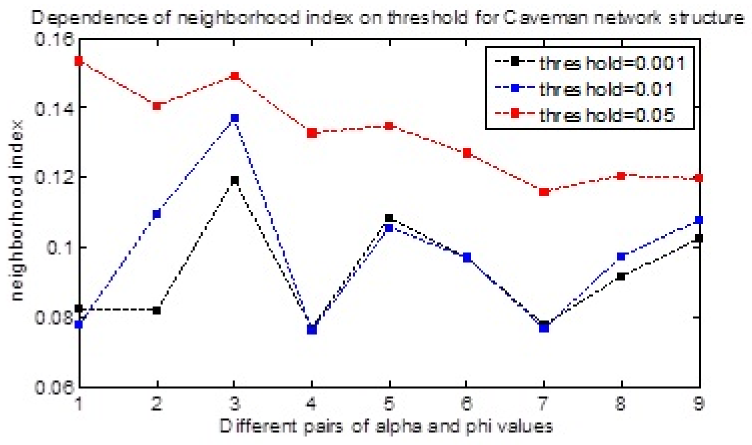


Figure 7: Dependence of neighborhood index on complex contagion threshold. Horizontal axis corresponds to distinct pairs of alpha-phi and vertical axis is neighborhood index. Each curve corresponds to a different value of complex contagion threshold. Red curve corresponds to the largest value of complex contagion threshold which has the largest value of neighborhood index while black curve corresponds to the smallest value of complex contagion threshold and it has the smallest neighborhood index.

6.1.2 Frequency of Dominance

This feature is interesting if one considers scientific society. How dominant are dominant ideas in the scientific community? Does the structure of the community influence this dominance? For all parameter combinations and for all network structures in our simulations, the frequency of dominance of ideas increased with time. This may suggest that none of these structures impede the adoption of new ideas. More surprisingly, however, the increase in dominance frequency progressed more quickly for the caveman network structure (Figure 8). Could it be that this structure encourages nodes to adopt dominant ideas more easily? This may not be generalizable because the results are quite sensitive to parameter values due to the stochasticity of the simulations. For example, Figure 8 shows a different combination of parameters, and here the faster increase in the dominance is not observed for the caveman structure.

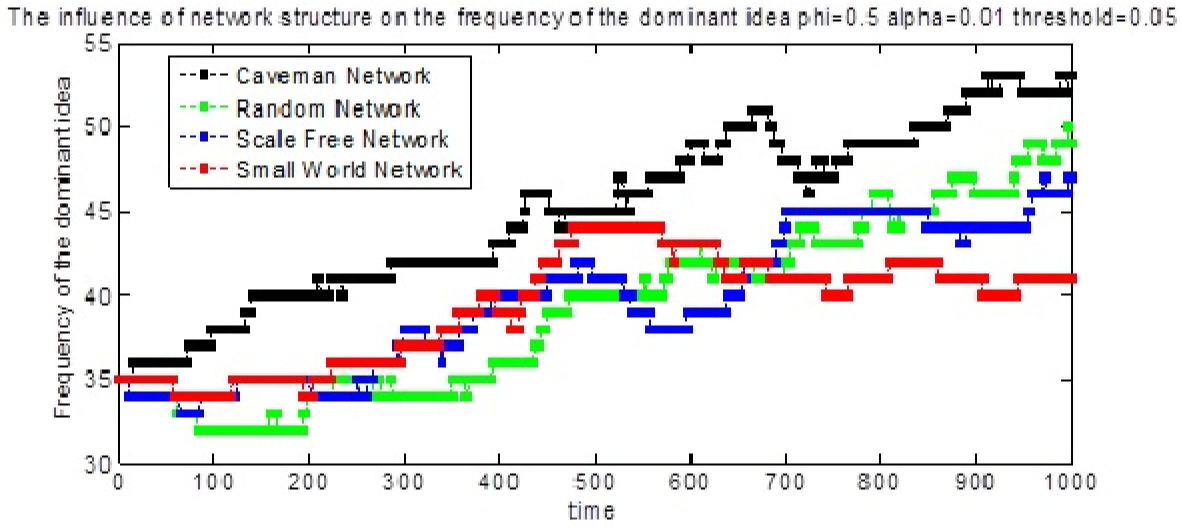
6.2 Effects of Idea Distribution on Network Structure

We observed that the results of applying a parallel idea distribution on the features of the network structure were quite different to the results of the random and anti-parallel idea distributions. It is interesting that for all parameter combinations and for each idea distribution, the network always remained fully connected. This could be because of the nature of our model: in order to disconnect with one node, there must be another node with the same idea to connect with. The nodes in a caveman network structure are, in a sense, 'saturated' since they are fully connected to their caves, and thus they remain connected to at least one of their original cave members.

While the networks always remained fully connected (Figure 9), the remaining features changed. These effects did not change with the parameter α for any of the idea distributions (see Figure 10, Figure 11 and Figure 12). Given these simulation results, perhaps a caveman-structured scientific community would also manage certain levels of innovativity without changing its fundamental structure.

6.2.1 Clustering Coefficient

The clustering coefficient of resulting networks varied with the type of idea distribution. When nodes in the same cave shared the same idea (parallel distribution), the clustering coefficient was larger (Figure 13). Additionally, the clustering coefficients for both the random and the anti-parallel idea distributions decreased in a similar manner regardless of the parameter combinations. Both of these results are intuitive since less rewiring would have taken place for the parallel distribution case, and the high clustering coefficient of the caveman structure would have been conserved,



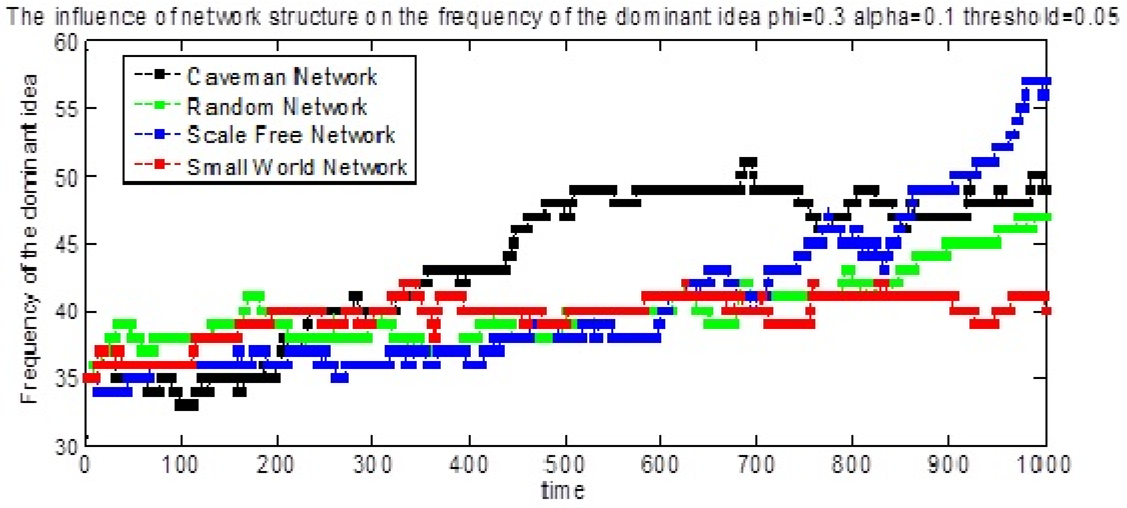


Figure 8: Frequency of the dominant idea at each time for four distinct network structures and two different sets of parameters: phi=0.5, alpha=0.01, threshold=0.05 (top) and phi=0.3, alpha=0.1, threshold=0.05 (bottom).

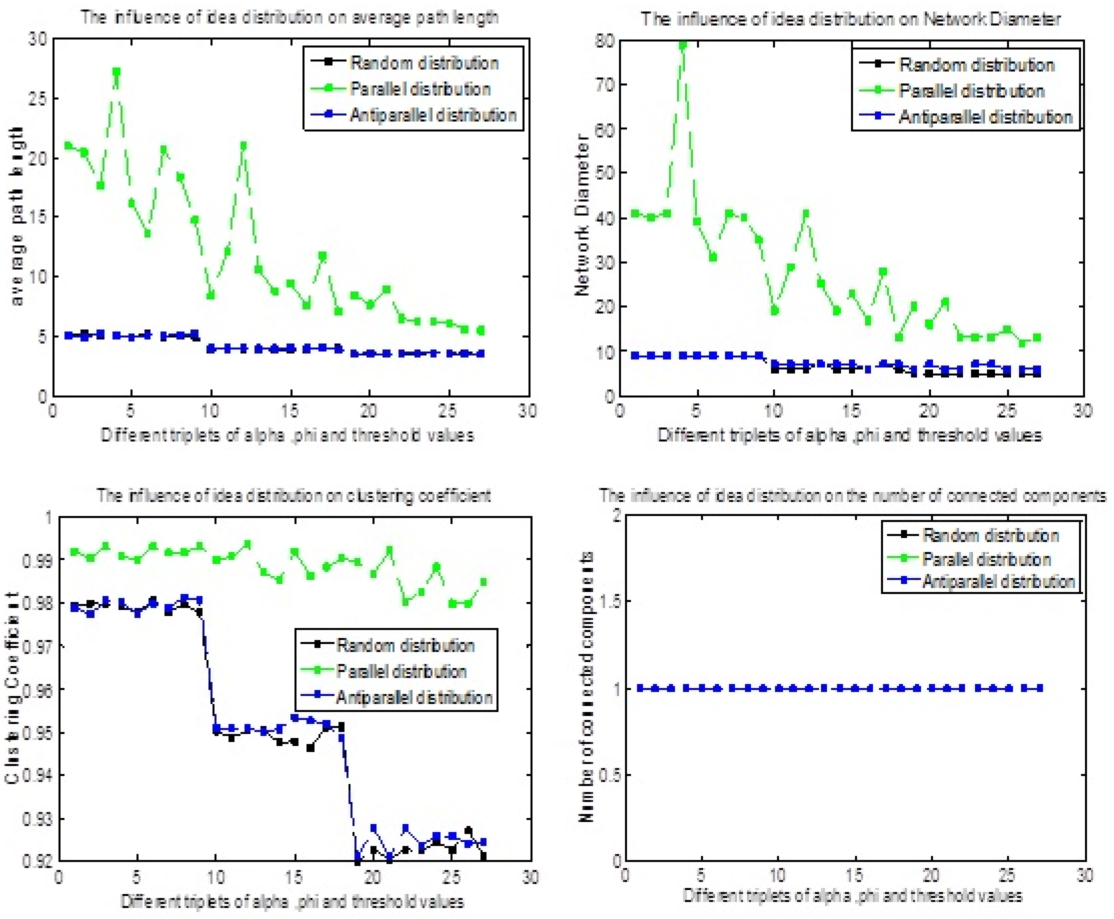


Figure 9: The influence of idea distribution on average path length (upper left), network diameter (upper right), clustering coefficient (bottom left) and number of connected components (bottom right) for 27 distinct sets of parameters (triplets of alpha-phi-threshold).

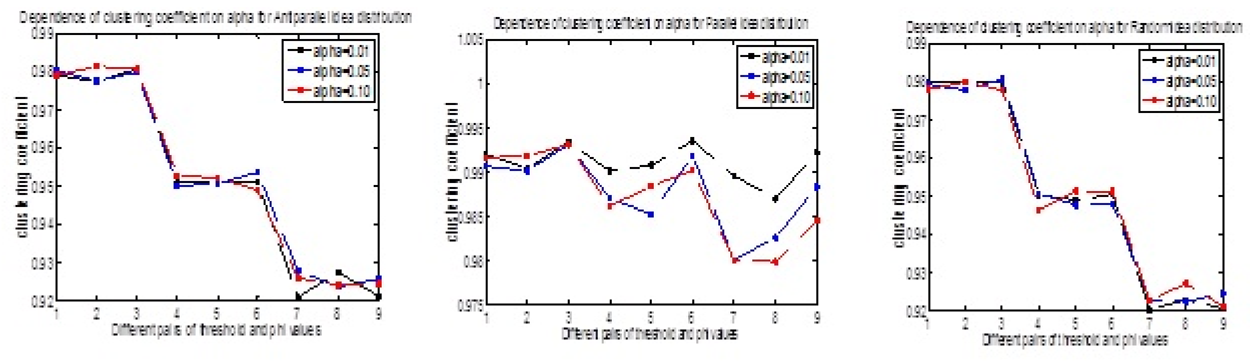


Figure 10: Dependence of clustering coefficient on alpha for each idea distribution. Horizontal axis corresponds to distinct pairs of phi-threshold and vertical axis is the clustering coefficient. Each curve corresponds to a different value of alpha. There is no dependence on alpha for clustering coefficient of the networks with any idea distribution.

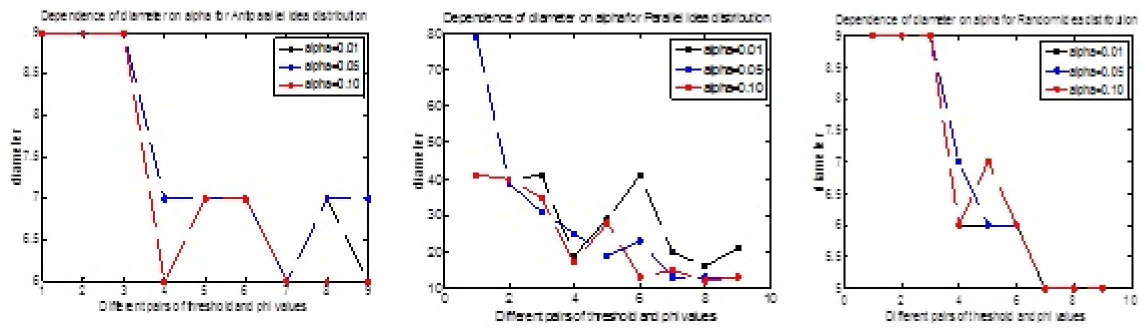


Figure 11: Dependence of network diameter on alpha for each idea distribution. Horizontal axis corresponds to distinct pairs of phi-threshold and vertical axis is the network diameter. Each curve corresponds to a different value of alpha. There is no dependence on alpha for the diameters of the networks with any idea distribution

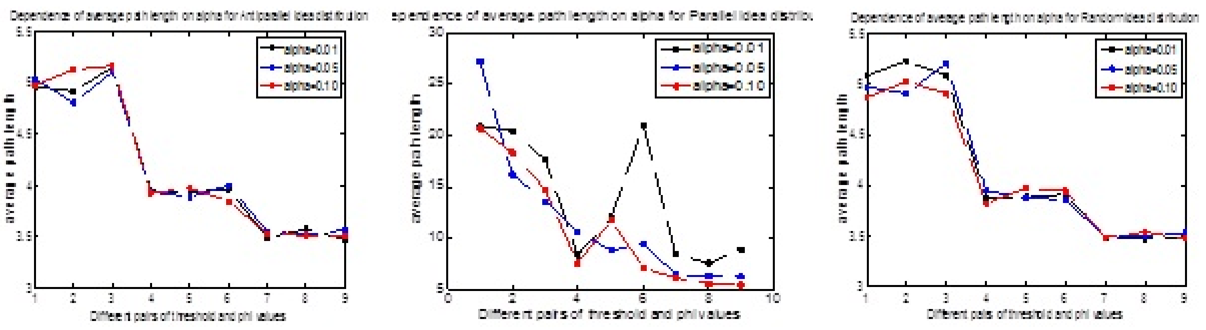


Figure 12: Dependence of average path length on alpha for each idea distribution. Horizontal axis corresponds to distinct pairs of phi-threshold and vertical axis is the average path length. Each curve corresponds to a different value of alpha. There is no dependence on alpha for average path length of the networks with any idea distribution.

whereas more rewiring would have ocurred for the two other distributions, thus decreasing the clustering coefficients.

Dependence on parameters Increasing values of the rewiring parameter ϕ decreased the clustering coefficient for all starting distributions, especially for the random and anti-parallel distributions (Figure 14). This again is intuitive since ϕ increases the chances of changing connections in a highly clustered network. Increasing values of δ, however, only slightly increased the clustering coefficient when using a parallel idea distribution (Figure 15).

6.2.2 Average Path Length

The average path length of resulting networks was larger when a parallel idea distribution was used (Figure 9). This is not surprising, since the structure of the caveman network was more preserved (because most nodes were already connected to like-minded nodes and did not need to rewire), and its average path length is larger than that of more randomized networks, such as the ones resulting from a larger amount of rewiring.

Dependence on parameters Increasing ϕ, regardless of the idea distribution, decreased the average path length. This effect was more prononced for the random and anti-parallel idea distribution cases (Figure 16). This is another intuitive result since more rewiring naturally disturbs the rigid structure of a caveman network,

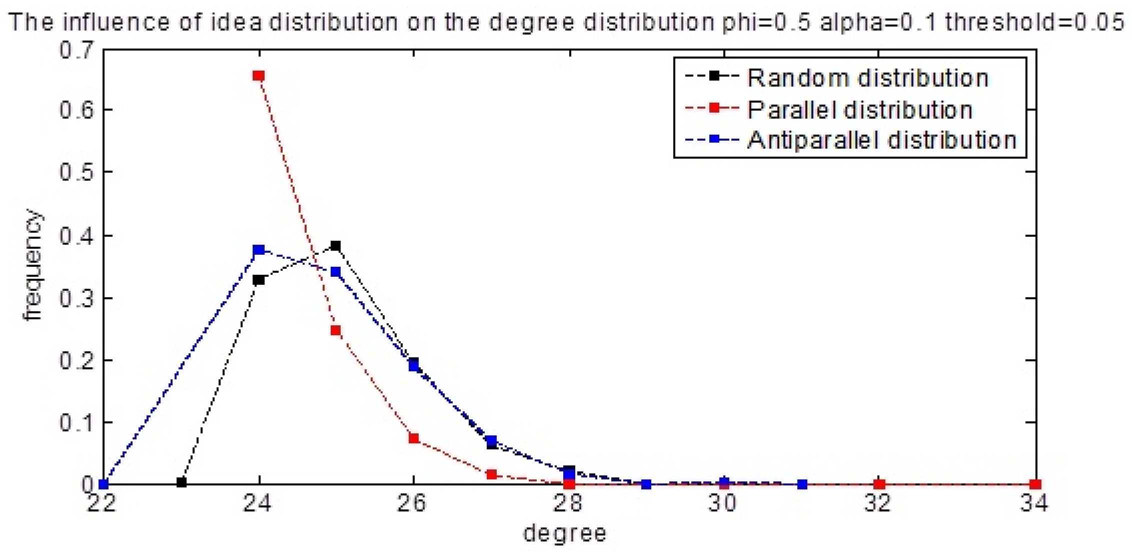


Figure 13: The influence of idea distribution on the degree distribution of the network. Horizontal axis corresponds to the degree of the nodes and vertical axis corresponds to the relative frequency of the nodes with that certain degree.

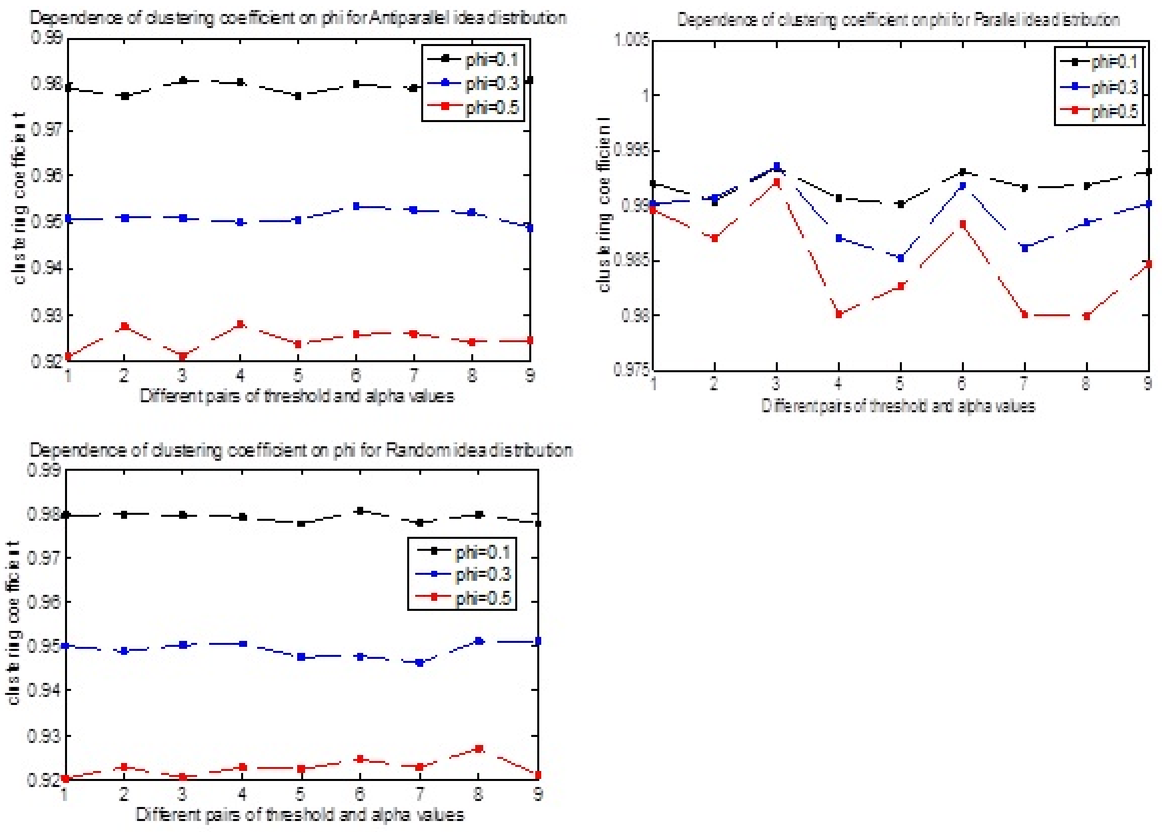
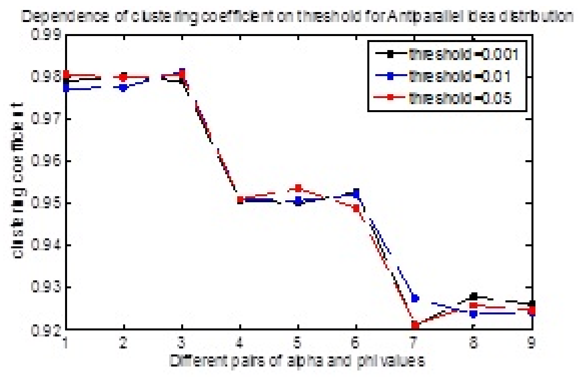
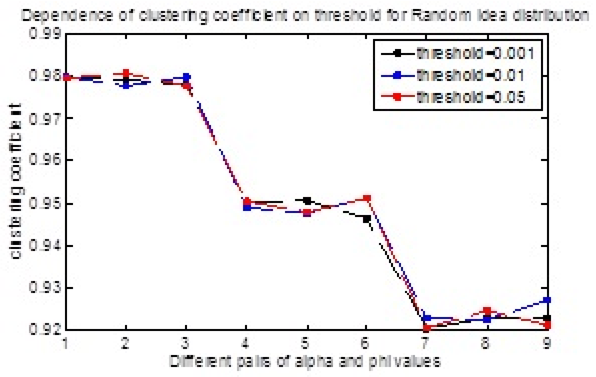


Figure 14: Dependence of clustering coefficient on phi for each idea distribution. Horizontal axis corresponds to distinct pairs of alpha- threshold and vertical axis is the clustering coefficient. Each curve corresponds to a different value of phi.





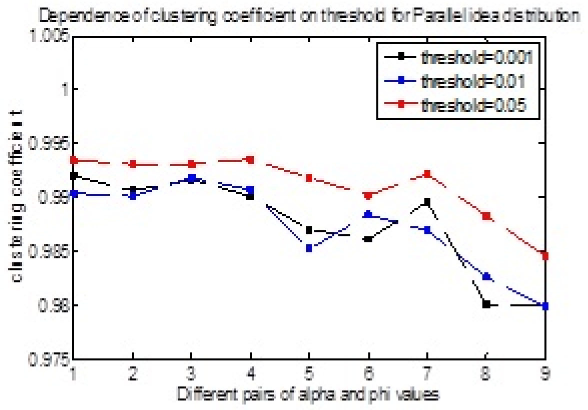


Figure 15: Dependence of clustering coefficient on complex contagion threshold for each idea distribution. Horizontal axis corresponds to distinct pairs of phi-alpha and vertical axis is the clustering coefficient. Each curve corresponds to a different value of complex contagion threshold.

thereby decreasing its large average path length; rewiring also occurs less frequently in the parallel idea distribution case because most nodes are already connected to like-minded nodes. Changing values of δ (which requires more than one neighbouring node to have the same idea in order to influence the chosen node) did not change the effects of idea distributions on the average path length (Figure 17). This is natural, since nodes were either already connected to a significant number of nodes with the same idea (in the case of the parallel distribution), not connected to any other node with the same idea (the anti-parallel case), or connected to nodes with random assignment of ideas. Thus the threshold δ would either already be fulfilled from the start, would definitely not be fulfilled, or would have a very small chance of being fulfilled, respectively.

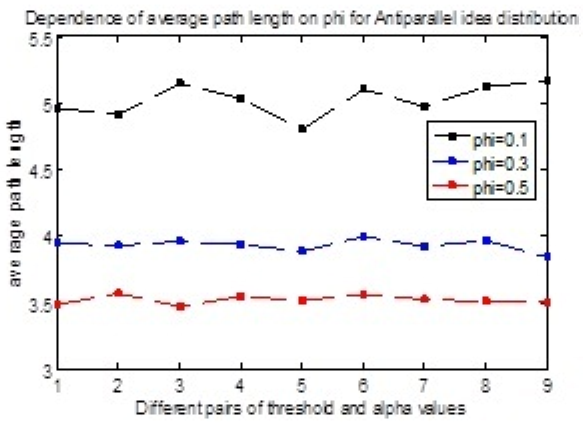
6.2.3 Network Diameter

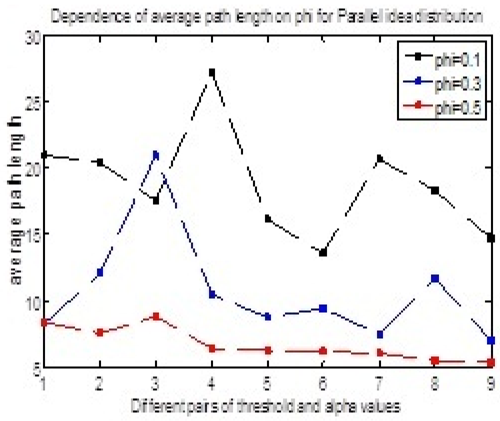
Similar to the clustering coefficient and the average path length, the network diameter was larger when the parallel idea distribution was applied (Figure 9). This distribution encouraged the structure of the caveman network to remain mostly unchanged, and thus the farthest distance between two nodes was larger than for the case of random or anti-parallel distributions, where more rewiring occured.

Dependence on parameters As ϕ increased, the network diameter decreased for all idea distributions, and slightly less for the parallel distribution (Figure 19). Rewiring a caveman network intuitively may decrease the network diameter by connecting more of its caves. Similar to the average path length, values of δ did not change the behaviour of the network diameter given the idea distributions (Figure 18).

6.2.4 Degree Distribution

Random graphs have a somewhat normally distributed degree distribution. Scale-free graphs, on the other hand, have a degree distribution that follows a scale-free power-law. We observed that the degree distribution of the networks depended on the idea distribution (Figure 13). A parallel idea distribution resulted in a degree distribution similar to that of a scale-free graph, which is not surprising. Caveman graphs have two or three different degrees for their nodes, and allowing for some rewiring would 'smooth' out this discrete distribution. Similar to previous results, the random and anti-parallel idea distributions behaved similarly: their degree distribution was similar to that of random graphs. It is interesting to see such a visible difference in these distributions over relatively few time steps (1000 steps), regardless of the parameter combinations.





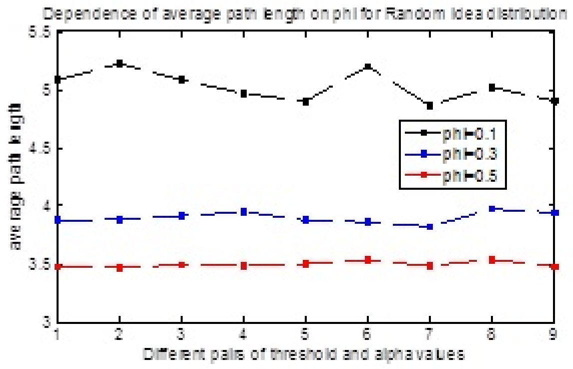


Figure 16: Dependence of average path length on phi for each idea distribution. Horizontal axis corresponds to distinct pairs of alpha-threshold and vertical axis is the average path length. Each curve corresponds to a different value of phi.

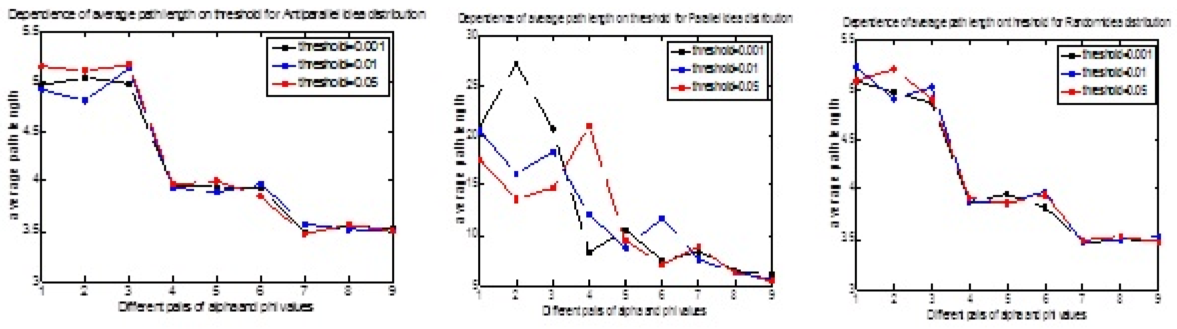


Figure 17: Dependence of average path length on complex contagion threshold for each idea distribution. Horizontal axis corresponds to distinct pairs of alpha-phi and vertical axis is the average path length. Each curve corresponds to a different value of complex contagion threshold. There is no dependence on complex contagion for average path length with any idea distribution.

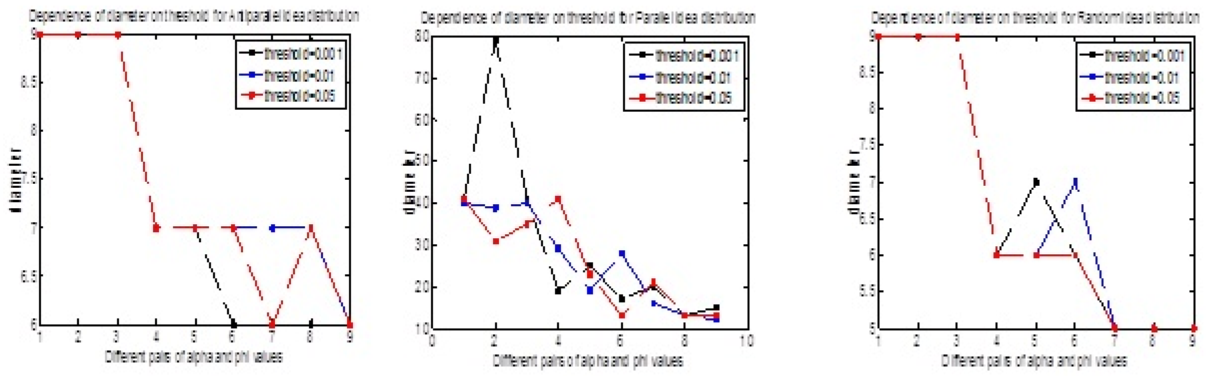


Figure 18: Dependence of network diameter on complex contagion threshold for each idea distribution. Horizontal axis corresponds to distinct pairs of phi-alpha and vertical axis is the network diameter. Each curve corresponds to a different value of complex contagion threshold. There is no dependence on complex contagion for network diameter with any idea distribution.

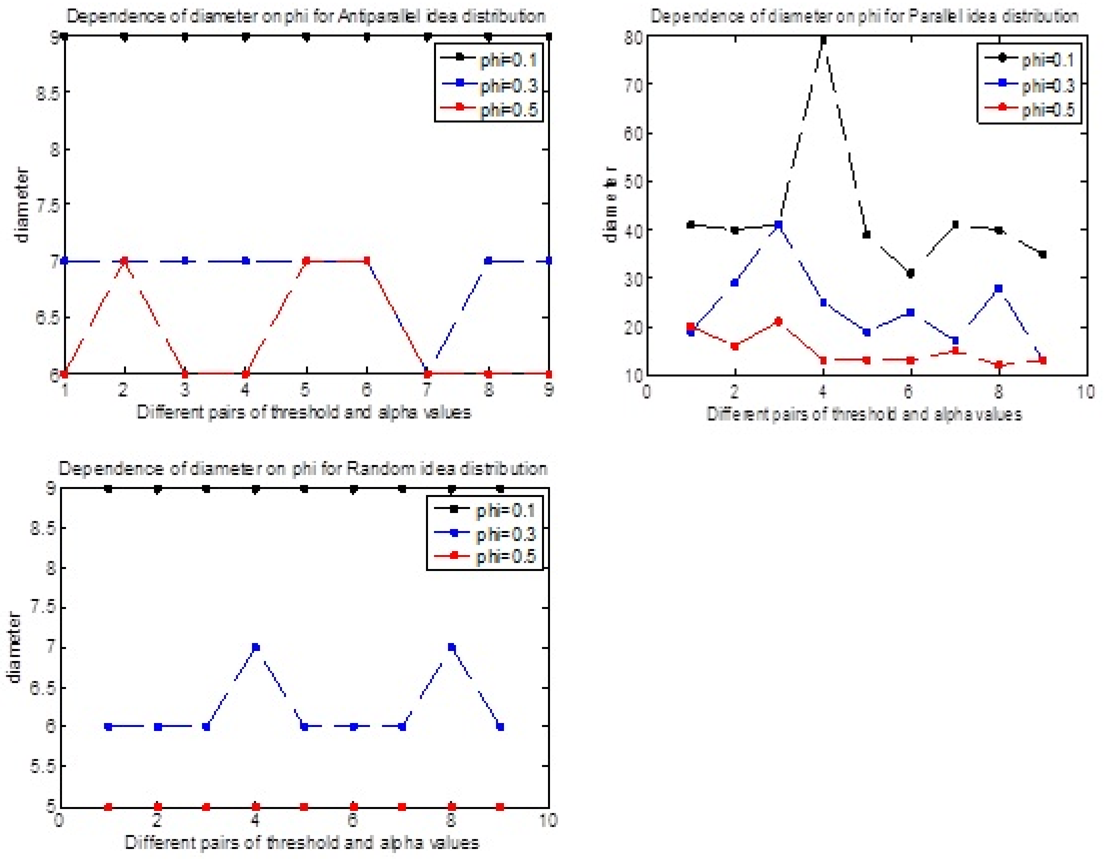


Figure 19: Dependence of network diameter on phi for each idea distribution. Horizontal axis corresponds to distinct pairs of alpha-threshold and vertical axis is the network diameter. Each curve corresponds to a different value of phi.

6.3 Discussion

As previously mentioned, we observed that some features of the idea distribution varied with the type of network structure. We also observed that some features of the network structure varied with the kind of idea distribution implemented. Some of these results were intuitive and expected, while others were not.

Network structure mainly influenced two features of the idea distribution: the intra-idea distance and the neighbourhood index. The intra-idea distance was larger for the caveman and small world networks, as anticipated. However, contrary to expectations, the neighbourhood index was larger for these two structures, and the caveman structure did not decrease the frequency of dominance (and neither did any other structure). Additionally, the values of ϕ and δ did correlate with changes in the values of the intra-idea distance (by decreasing it) and the neighbourhood index (with different effects depending on the structure), but not always: δ did not vary the results of the intra-idea distance or the neighbourhood index of the random and scale-free networks, and ϕ did not vary the neighbourhood index in the small world network. Lastly, and also contrary to expectations, increasing the α values increased the neighbourhood index of networks.

Four of the five features of the network structure changed with different idea distributions. The connected component interestingly remained one, regardless of the idea distribution and parameter values. Feature values of the anti-parallel and random idea distributions differed from those of the parallel distribution. Most features changed as expected: the clustering coefficient, average path length, and network diameter were larger when the parallel idea distribution was used as compared to the other two, and their values were similar to those of a caveman network structure. The clustering coefficient of the anti-parallel idea distribution networks did decrease, as expected. The degree distribution did increase in variance (becoming more similar to a normal distribution) for the networks that were initiated with a random or anti-parallel idea distribution, but the degree distribution of the parallel idea distribution networks resembled that of a scale-free power law. Values of ϕ influenced the features of the random and antiparallel distribution networks the most. However, the structure features of the networks with the random idea distribution were not more sensitive to parameter values as would have been expected; they behaved similar to those of the networks with the anti-parallel idea distribution. Values of δ only correlated with changes in the clustering coefficient of the networks with a parallel idea distribution, and only weakly. Values of α did not correlated with any changes of feature values.

7 Summary and Outlook

To conclude our simulation study, our results support the general idea that network structure and qualities of the ideas held in them may mutually influence each other. The more rigid the structure of a network was, the more likely that the intra-idea distance and neighbourhood index of the network was larger. These two features varied more with the innovation rate α for less structured networks, and varied more with the complex contagion threhold δ for the most rigid structure - the caveman network. The average dominance time seemed to vary more with parameter values (ϕ, δ, and α) than with the network structures, whereas the frequency of dominance of ideas did not decrease on average in the long run with any network structure.

A network in which the pattern of ideas held by the nodes are 'in accord' within the clusters of the caveman network maintained more of the structure features of a caveman network. These values varied with values of rewiring probabilities ϕ and complex contagion threholds δ. Networks with a random pattern of ideas or with a pattern with more 'disaccord' within the clusters resulted in a structure that began to resemble a random graph more than a caveman network. These networks' values varied more with the parameter ϕ of rewiring. Nodes given the chance to rewire with nodes that share the same idea had more opportunity to do so in the random and 'disaccord' idea patterns than in the pattern which already had much accord.

Thus, in addition to the structure of networks and the pattern of ideas in them, complex contagion thresholds, innovation rates, and the probability of creating new connections (while reflecting 'preferences' of being connected with like-minded others) tend to influence features of the network. Several extensions to the proposed model could be investigated to better understand the relationships between network structure and idea distribution.

Rewiring criteria Future simulations may compare the emerging features of idea distributions not just between network structures, but also between different rewiring criteria. It is not clear how much of the features of the idea distribution in this simulation study was a result of the network structure or of the 'preference' that nodes had in rewiring to like-minded nodes. Therefore, these results could be compared to (1) random rewiring or, to allow structure to play a larger role, (2) to allow random rewiring to nodes that are at most a distance of three nodes away. This is somewhat more realistic since most people connect with individuals who are somewhat in their vicinity through mutual connections.

Complex contagion and innovation Considering complex contagion, novel ideas in our model were at a disadvantage for spreading in the network. Allowing novel ideas to have a larger influence weight may be one way to grant the ability of them to spread to other nodes. Alternatively, the complex contagion threshold for novel ideas may be lowered.

Random idea adoption Future simulations may investigate the effects of network structure on idea distributions by comparing to a 'benchmark' model. This model would update the ideas of nodes at random, and thus the effects of connections may be better observed within network structures and then compared between network structures.

8 References

A MATLAB code

A.1 Main script

```
%% The first phase: Simulation to study the influence of network ...
    structure on the opinion%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% The following parameters will remain constant during this study,
%%% so we won't play with them %%%
n=1000; %% the number of agents %%%
m=40; %% initial number of clusters for caveman matrix%%%
p=40; %% initial number of opinions%%
t_end=1000; %%% number of iterations%%
%%% for the following parameters we'll run different simulations based on
%%% conmbinatorial complexity of the parameters%%%
phi_choices=[0.1,0.3,0.5]; %%% network reorganization rate%%
alpha_choices=[0.01,0.05,0.10]; %%% innovation rate %%%
threshold_choices=[0.001,0.01,0.05];%%% threshold for complex contagion %%%


%%% a totally random idea distribution , independent of the connectivity
%%% matrix, so applicable for every network structure is defined for this
%%% phase %%%
vec1=zeros(1,n);
for i=1:n
    vec1(i)=ceil(rand()*p);
end

for choice1=1:4
%% Step1: Definition of Different initial matrices ...
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%% For each simulation, we must choose one of the ...
        following connectivity matrices %%%%%%%%%%%%%%%%%%%%%%%
    switch choice1
        case 1
         %%% option 1: Caveman Connectivity Matrix
         mat1=step1_caveman(n,m);
         s1='Caveman';
        case 2
        %%% option 2: Random Connectivity Matrix
        prob=0.025; %%% probability of edge formation between any pairs ...
            of edges
        mat1=step1_randomgraph(n,prob);
        s1='Random';
        case 3
        %%% option 3: Scale Free Connectivity Matrix
        m0=24; % number of initially placed nodes
        m1=12; % number of nodes a new added node is connected to, 1 ≤ ...
            m1 < m0
        mat1=step1_scalefree(n, m0, m1);
        s1='Scale_free';
        case 4
        %%% option 4: Small world Connectivity Matrix
        ka=24; %% mean degree (assumed to be an even integer)
        beta=0.01; %% rewiring probability
        mat1= step1_smallworld(n, ka, beta);
        s1='Small_world';
    end

    %% Step2: Simulation ...
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for choice2=1:3
        phi=phi_choices(choice2);
        for choice3=1:3
            alpha=alpha_choices(choice3);
           for choice4=1:3
             threshold=threshold_choices(choice4);

             [mat2,vec2,dominant_freq,most_freq]=step2(t_end,phi,alpha,mat1,vec1,p,threshold);
                 %%% obtaining the final matrix and vector after running ...
                 simulation.

             %%%% We need step4c here, since it's outputs will be the input
             %%%% for step 3b
             %%%%%%% step4c: number of connected components %%%%%%%%%%
             sp_mat2=sparse(mat2);
             [s,c]=graphconncomp(sp_mat2); %% matlab built in function ...
                 'Bioinformatics Toolbox'
             %%% s: number of connected components
             %%% c: vector which assigns each node to a connected component

             %% Step3: Results for structure to idea ...
                 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

             %%%%%%% step3a: defining the average intra_idea ...
                 neighbourhood index %%%%%%%
             neighbor_index=step3a(mat2,vec2);

             %%%%%%% step3b: defining the average intra_idea distance ...
                 %%%%%%%
             intra_idea_distance=step3b(mat2,vec2,s,c);

             %%%%%%% step3c: frequency of dominant idea with respect to ...
                 time %%%%%%%
             %%% is the third output of the step2 function ...
                 (dominant_freq)%%%
```

```
                dominant_freq;

                %%%%%%% step3d: Fraction of novel ideas (novelity index) %%%%%%
                nov_index=(length(find(vec2>p)))/(length(vec2)); %%% ...
                    indicates the fraction of agents holding the newly ...
                    generated ideas

                %%%%%%% step3e: defining the average dominance time (the ...
                    average amount of time in which the dominating idea ...
                    keeps it's dominance over diffrent dominance periods)
                average_dominance_time=step3e(most_freq);

                %%% naming the file which saves the results
                s2=int2str(choice2);
                s3=int2str(choice3);
                s4=int2str(choice4);
                name=['phase1_',s1,'_',s2,'_',s3,'_',s4];
                save(name);
              end
           end
       end
end

clear;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% The second phase: Simulation to study the influence of opinion on the ...
    network structure%%%%%%%%%%%%%%%%

%%% The following three parameters will remain constant during this study,
%%% so won't play with them %%%
n=1200; %% the number of agents
m=40; %% initial number of clusters for caveman matrix
p=40; %% initial number of opinions
t_end=1000; %%% number of iterations%%
%%% for the following parameters we'll run different simulations based on
%%% conmbinatorial complexity of the parameters%%%
phi_choices=[0.1,0.3,0.5]; %%% network reorganization rate%%
alpha_choices=[0.01,0.05,0.10]; %%% innovation rate %%%
threshold_choices=[0.001,0.01,0.05];%%% threshold for complex contagion %%%



%%%%%in this phase we'll keep connectivity matrix constant, so we only use
%%%%%Caveman connectivity matrix %%%
mat1=step1_caveman(n,m);


for choice1=1:3
```

```
%% Step1: Definition of Different initial idea vectors ...
    %%%%%%%%%%%%%%%%%%%%%%%
    switch choice1
           case 1
           %%%%%% option1: Random idea vector %%%%%%%
           %%% a totally random idea distribution , independent of the ...
               connectivity
           %%%% matrix, so applicable for every network structure
           vec1=zeros(1,n);
           for i=1:n
                vec1(i)=ceil(rand()*p);
           end
           s1='Random';
           case 2
           %%%%%% option2: Parallel idea vector %%%%%%%
           %%% This idea vector is applicable only for caveman ...
               connectivity matrix in
           %%% which every agents inside a cluster have the same idea
           vec1=zeros(1,n);
           for i=1:(m-1) %%% for each cluster
              for j=1:ceil(n/m)
                  vec1((i*ceil(n/m))+j)=i; %%% all agents will hold the ...
                      i-th idea
              end
           end
           s1='Parallel';
           case 3
           %%%%%% option3: Antiparallel idea vector %%%%%%%
           %%% This idea vector is applicable only for caveman ...
               connectivity matrix in
           %%% which every agents inside a cluster have different idea
           vec1=zeros(1,n);
           for i=1:(m-1) %%% for each cluster
               for j=1:ceil(n/m)
                  vec1((i*ceil(n/m))+j)=j; %%% all agents will hold ...
                      different idea
               end
           end
           s1='Antiparallel';
    end
    %% Step2: Simulation ...
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for choice2=1:3
        phi=phi_choices(choice2);

        for choice3=1:3
            alpha=alpha_choices(choice3);
          for choice4=1:3
            threshold=threshold_choices(choice4);

```

```
                [mat2,vec2,dominant_freq,most_freq]=step2(t_end,phi,alpha,mat1,vec1,p,threshold);
                    %%% obtaining the final matrix and vector after running ...
                    simulation.


                %% Step3: Results for idea to structure ...
                    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                %%%%%%% step4a: clustering coefficient of the final network ...
                    %%%%%%%%
                clust_coefficient=step4a(mat2);

                %%%%%%% step4b: degree distribution of the final network ...
                    %%%%%%%%%%%
                [dgr,frq]=step4b(mat2);
                average_degree=sum(dgr.*frq)/sum(frq);

                %%%%%%% step4c: number of connected components %%%%%%%%%%
                sp_mat2=sparse(mat2);
                [s,c]=graphconncomp(sp_mat2); %% matlab built in function ...
                    'Bioinformatics Toolbox'
                %%% s: number of connected components
                %%% c: vector which assigns each node to a connected component

                %%%%%%% step4d: average path length for the final network ...
                    %%%%%%%%%%
                average_path_length = step4d( mat2,s,c );

                %%%%%%% step4e: Diameter of the network %%%%%%%%%%%%%%%%%%
                diam=step4e(mat2,s,c);

                %%%naming and saving
                s2=int2str(choice2);
                s3=int2str(choice3);
                s4=int2str(choice4);
                name=['phase2_',s1,'_',s2,'_',s3,'_',s4];
                save(name);
            end
        end
    end
end

clear;
```

A.2 Step 1: generation of structure networks

randomgraph.m

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% code adopted from %%%
%%% Modeling and Simulating Social Systems with MATLAB %%%
%%% http://www.soms.ethz.ch/teaching/MatlabFall2012 %%%
%%% Authors: Stefan Brugger and Cristoph Schwirzer, 2011 %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function  final= step1_randomgraph(n, p)
% Generates an undirected random graph (without self-loops) of size n (as
% described in the Erdoes-Renyi model)
%
% INPUT
% n: [1]: number of nodes
% p: [1]: probability that node i and node j, i != j, are connected by ...
    an edge
%
% OUTPUT
% final: [n n] full symmetric adjacency matrix representing the ...
    generated graph

% Note: A generation based on sprandsym(n, p) failed (for some values of p
% sprandsym was far off from the expected number of n*n*p non-zeros), ...
    therefore
% this longish implementation instead of just doing the following:
%
% B = sprandsym(n, p);
% A = (B-diag(diag(B))≠0);
%

% Idea: first generate the number of non-zero values in every row for a ...
    general
% 0-1-adjacency matrix. For every row this number is distributed ...
    binomially with
% parameters n and p.
%
% The following lines calculate "rowsize = binoinv(rand(1, n), n, p)", ...
    just in a
% faster way for large values of n.

% generate a vector of n values chosen u.a.r. from (0,1)
v = rand(1, n);
% Sort them and calculate the binomial cumulative distribution function with
% parameters n and p at values 0 to n. Afterwards match the sorted random
% 0-1-values to those cdf-values, i.e. associate a binomial distributed ...
    value
```

```
% with each value in r. Each value in v also corresponds to a value in r:
% permute the values in rowSize s.t. they correspond to the order given ...
    in v.
[r index] = sort(v); % i.e. v(index) == r holds
rowSize = zeros(1, n);
j = 0;
binoCDF = cumsum(binopdf(0:n, n, p));
for i = 1:n
    while j<n && binoCDF(j+1)<r(i)
        j = j + 1;
    end
    rowSize(i) = j;
end
rowSize(index) = rowSize;

% for every row choose the non-zero entries in it
nNZ = sum(rowSize);
I = zeros(1, nNZ);
J = zeros(1, nNZ);
j = 1;
for i = 1:n
    I(j:j+rowSize(i)-1) = i;
    J(j:j+rowSize(i)-1) = randsample(n, rowSize(i));
    j = j + rowSize(i);
end

% restrict I and J to indices that correspond to entries above the main ...
    diagonal
% and finally construct a symmetric sparse matrix using I and J
upperTriu = find(I<J);
I = I(upperTriu);
J = J(upperTriu);
A = sparse([I;J], [J;I], ones(1, 2*size(I, 2)), n, n);
final=full(A);
end % random_graph(...)
```

scalefree.m

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% code adopted from                                     %%%
%%% Modeling and Simulating Social Systems with MATLAB     %%%
%%% http://www.soms.ethz.ch/teaching/MatlabFall2012        %%%
%%% Authors: Stefan Brugger and Cristoph Schwirzer, 2011   %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function A = scalefree(n, m0, m)
% Use the Barabasi-Albert model to generate a scale free graph of size n (as
% described in Albert-Laszlo Barabasi & Reka Albert: "Emergence of scaling
% in random networks")
%
% INPUT
% n: [1]: number of nodes
% m0: [1]: number of initially placed nodes
% m: [1]: number of nodes a new added node is connected to, 1 ≤ m < m0
%
% OUPUT
% A: [n n] sparse symmetric adjacency matrix representing the generated ...
    graph

% Start with a graph of size m0 and add edges to this graph. Each of ...
    these m0
% nodes is connected to at least m nodes.
B = zeros(m0, m0);
for i = 1:m0
    neighbors = randsample(m0-1, m);
    neighbors = neighbors + (neighbors≥i);
    B(i,neighbors) = 1;
    B(neighbors,i) = 1;
end

% Create a vector of edges added so far, i.e. nodes edge(2*i) and ...
    edge(2*i-1),
% 1 ≤ i ≤ nEdges, are connected by an edge.
[rows, columns] = find(triu(B));
nEdges = size(rows, 1);
edges = reshape([rows';columns'], 2*nEdges, 1);
edges = [edges; zeros(2*(n-m0)*m,1)];

% Add nodes m0+1:n, one at a time. Each node is connected to m existing ...
    nodes,
% where each of the existing nodes is chosen with a probability that is
% proportional to the number of nodes it is already connected to.
used = zeros(n, 1); % is a node already used in a timestep?
for i = m0+1:n
    neighbors = zeros(1, m);
    for j=1:m
       k = edges(randi(2*nEdges));
       while used(k)
           k = edges(randi(2*nEdges));
       end
       used(k) = 1;
       neighbors(j) = k;
    end
    used(neighbors) = 0;
    edges(2*nEdges+1:2*nEdges+2*m) = reshape([repmat(i, 1, m); ...
        neighbors], ...
     1, 2*m);
    nEdges = nEdges+m;
end

% finally construct a symmetric adjacency matrix using the vector of edges
edges = edges(1:2*nEdges);
first = edges(1:2:end);
second = edges(2:2:end);
A = sparse([first;second], [second;first], ones(2*nEdges, 1), n, n);

end % scale_free(...)
```

smallworld.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% code adopted from                                      %%%
%%% Modeling and Simulating Social Systems with MATLAB     %%%
%%% http://www.soms.ethz.ch/teaching/MatlabFall2012        %%%
%%% Authors: Stefan Brugger and Cristoph Schwirzer, 2011   %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function A = smallworld(n, k, beta)
% Generate a small world graph using the "Watts and Strogatz model" as
% described in Watts, D.J.; Strogatz, S.H.: "Collective dynamics of
% 'small-world' networks."
% A graph with n*k/2 edges is constructed, i.e. the nodal degree is n*k for
% every node.
%
% INPUT
% n: [1]: number of nodes of the graph to be generated
% k: [1]: mean degree (assumed to be an even integer)
% beta: [1]: rewiring probability
%
% OUPUT
% A: [n n] sparse symmetric adjacency matrix representing the generated ...
    graph
```

```
% Construct a regular lattice: a graph with n nodes, each connected to k
% neighbors, k/2 on each side.
kHalf = k/2;
rows = reshape(repmat([1:n]', 1, k), n*k, 1);
columns = rows+reshape(repmat([[1:kHalf] [n-kHalf:n-1]], n, 1), n*k, 1);
columns = mod(columns-1, n) + 1;
B = sparse(rows, columns, ones(n*k, 1));
A = sparse([], [], [], n, n);

% With probability beta rewire an edge avoiding loops and link duplication.
% Until step i, only the columns 1:i are generated making implicit use ...
    of A's
% symmetry.
for i = [1:n]
    % The i-th column is stored full for fast access inside the ...
        following loop.
    col= [full(A(i, 1:i-1))'; full(B(i:end, i))];
    for j = i+find(col(i+1:end))'
        if (rand()<beta)
            col(j)=0;
            k = randi(n);
            while k==i || col(k)==1
                k = randi(n);
            end
            col(k) = 1;
        end
    end
    A(:,i) = col;
end

% A is not yet symmetric: to speed things up, an edge connecting i and ...
    j, i < j
% implies A(i,j)==1, A(i,j) might be zero.
T = triu(A);
A = T+T';

end % small_world(...)
```

step1_caveman.m

```
function cave_mat = step1_caveman(n,m)
%%%%%%%%%%%%  This function outputs a Caveman Matrix  %%%%%%%%%%%%%%%%
%%% n: the number of agents
%%% m: initial number of clusters
%%% p: initial maximum index of opinions
cave_mat=zeros(n,n); %Caveman Matrix

for i=1:n
    x=ceil(i/(n/m));
    for j=1:n
        y=ceil(j/(n/m));
        if x==y
            if i≠j
            cave_mat(i,j)=1; %definition of intracluster edges
            end
        end
    end
end
%%% x1 and x2 for each cluster represent the two agents who interact with
%%% nearby clusters%%
x1=zeros(1,m);
x2=zeros(1,m);
for i=1:m
    x1(i)=ceil(rand()*(n/m)+(n/m)*(i−1));
    ind=0;
    while(ind==0) %% This loop is used to prevent x1 and x2 to make the ...
        same numbers
        x2(i)=ceil(rand()*(n/m)+(n/m)*(i−1));
        if x2(i)≠x1(i)
            ind=1;
        end
    end
end
%%% definition of intercluster edges
for i=1:(m−1)
    cave_mat(x2(i),x1(i+1))=1;
    cave_mat(x1(i+1),x2(i))=1;
end
cave_mat(x1(1),x2(m))=1;
cave_mat(x2(m),x1(1))=1;
end
```

step1_randomgraph.m

```


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% code adopted from %%%
%%% Modeling and Simulating Social Systems with MATLAB %%%
%%% http://www.soms.ethz.ch/teaching/MatlabFall2012 %%%
%%% Authors: Stefan Brugger and Cristoph Schwirzer, 2011 %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function  final= step1_randomgraph(n, p)
% Generates an undirected random graph (without self-loops) of size n (as
% described in the Erdoes-Renyi model)
%
% INPUT
% n: [1]: number of nodes
% p: [1]: probability that node i and node j, i != j, are connected by ... an edge
%
% OUTPUT
% final: [n n] full symmetric adjacency matrix representing the ... generated graph

% Note: A generation based on sprandsym(n, p) failed (for some values of p
% sprandsym was far off from the expected number of n*n*p non-zeros), ... therefore
% this longish implementation instead of just doing the following:
%
% B = sprandsym(n, p);
% A = (B-diag(diag(B))≠0);
%

% Idea: first generate the number of non-zero values in every row for a ... general
% 0-1-adjacency matrix. For every row this number is distributed ... binomially with
% parameters n and p.
%
% The following lines calculate "rowsize = binoinv(rand(1, n), n, p)", ... just in a
% faster way for large values of n.

% generate a vector of n values chosen u.a.r. from (0,1)
v = rand(1, n);
% Sort them and calculate the binomial cumulative distribution function with
% parameters n and p at values 0 to n. Afterwards match the sorted random
% 0-1-values to those cdf-values, i.e. associate a binomial distributed ... value
% with each value in r. Each value in v also corresponds to a value in r:
% permute the values in rowSize s.t. they correspond to the order given ...
    in v.
[r index] = sort(v); % i.e. v(index) == r holds
rowSize = zeros(1, n);
j = 0;
binoCDF = cumsum(binopdf(0:n, n, p));
for i = 1:n
    while j<n && binoCDF(j+1)<r(i)
        j = j + 1;
    end
    rowSize(i) = j;
end
rowSize(index) = rowSize;

% for every row choose the non-zero entries in it
nNZ = sum(rowSize);
I = zeros(1, nNZ);
J = zeros(1, nNZ);
j = 1;
for i = 1:n
    I(j:j+rowSize(i)-1) = i;
    J(j:j+rowSize(i)-1) = randsample(n, rowSize(i));
    j = j + rowSize(i);
end

% restrict I and J to indices that correspond to entries above the main ...
    diagonal
% and finally construct a symmetric sparse matrix using I and J
upperTriu = find(I<J);
I = I(upperTriu);
J = J(upperTriu);
A = sparse([I;J], [J;I], ones(1, 2*size(I, 2)), n, n);
final=full(A);
end % random_graph(...)
```

step1_scalefree.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% code adopted from %%%
%%% Modeling and Simulating Social Systems with MATLAB %%%
%%% http://www.soms.ethz.ch/teaching/MatlabFall2012 %%%
%%% Authors: Stefan Brugger and Cristoph Schwirzer, 2011 %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function final = step1_scalefree(n, m0, m)
% Use the Barabasi−Albert model to generate a scale free graph of size n (as
% described in Albert−Laszlo Barabasi & Reka Albert: "Emergence of scaling
% in random networks")
%
% INPUT
% n: [1]: number of nodes
% m0: [1]: number of initially placed nodes
% m: [1]: number of nodes a new added node is connected to, 1 ≤ m < m0
%
% OUPUT
% final: [n n] full symmetric adjacency matrix representing the ...
    generated graph

% Start with a graph of size m0 and add edges to this graph. Each of ...
    these m0
% nodes is connected to at least m nodes.
B = zeros(m0, m0);
for i = 1:m0
    neighbors = randsample(m0−1, m);
    neighbors = neighbors + (neighbors≥i);
    B(i,neighbors) = 1;
    B(neighbors,i) = 1;
end

% Create a vector of edges added so far, i.e. nodes edge(2*i) and ...
    edge(2*i−1),
% 1 ≤ i ≤ nEdges, are connected by an edge.
[rows, columns] = find(triu(B));
nEdges = size(rows, 1);
edges = reshape([rows';columns'], 2*nEdges, 1);
edges = [edges; zeros(2*(n−m0)*m,1)];

% Add nodes m0+1:n, one at a time. Each node is connected to m existing ...
    nodes,
% where each of the existing nodes is chosen with a probability that is
% proportional to the number of nodes it is already connected to.
used = zeros(n, 1); % is a node already used in a timestep?
for i = m0+1:n
    neighbors = zeros(1, m);
    for j=1:m
       k = edges(randi(2*nEdges));
       while used(k)
           k = edges(randi(2*nEdges));
       end
       used(k) = 1;
       neighbors(j) = k;
    end
    used(neighbors) = 0;
    edges(2*nEdges+1:2*nEdges+2*m) = reshape([repmat(i, 1, m); ...
        neighbors], ...
     1, 2*m);
    nEdges = nEdges+m;
end

% finally construct a symmetric adjacency matrix using the vector of edges
edges = edges(1:2*nEdges);
first = edges(1:2:end);
second = edges(2:2:end);
A = sparse([first;second], [second;first], ones(2*nEdges, 1), n, n);
final=full(A);
end % scale_free(...)
```

step1_smallworld.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% code adopted from %%%
%%% Modeling and Simulating Social Systems with MATLAB %%%
%%% http://www.soms.ethz.ch/teaching/MatlabFall2012 %%%
%%% Authors: Stefan Brugger and Cristoph Schwirzer, 2011 %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function final = step1_smallworld(n, k, beta)
% Generate a small world graph using the "Watts and Strogatz model" as
% described in Watts, D.J.; Strogatz, S.H.: "Collective dynamics of
% 'small-world' networks."
% A graph with n*k/2 edges is constructed, i.e. the nodal degree is n*k for
% every node.
%
% INPUT
% n: [1]: number of nodes of the graph to be generated
% k: [1]: mean degree (assumed to be an even integer)
% beta: [1]: rewiring probability
%
% OUPUT
% A: [n n] sparse symmetric adjacency matrix representing the generated ...
    graph
```

```matlab

% Construct a regular lattice: a graph with n nodes, each connected to k
% neighbors, k/2 on each side.
kHalf = k/2;
rows = reshape(repmat((1:n)', 1, k), n*k, 1);
columns = rows+reshape(repmat([(1:kHalf) (n-kHalf:n-1)], n, 1), n*k, 1);
columns = mod(columns-1, n) + 1;
B = sparse(rows, columns, ones(n*k, 1));
A = sparse([], [], [], n, n);

% With probability beta rewire an edge avoiding loops and link duplication.
% Until step i, only the columns 1:i are generated making implicit use ...
    of A's
% symmetry.
for i = 1:n
    % The i-th column is stored full for fast access inside the ...
        following loop.
    col= [full(A(i, 1:i-1))'; full(B(i:end, i))];
    for j = i+find(col(i+1:end))'
        if (rand()<beta)
            col(j)=0;
            k = randi(n);
            while k==i || col(k)==1
                k = randi(n);
            end
            col(k) = 1;
        end
    end
    A(:,i) = col;
end

% A is not yet symmetric: to speed things up, an edge connecting i and ...
    j, i < j
% implies A(i,j)==1, A(i,j) might be zero.
T = triu(A);
A = T+T';
final=sparse(A);

end % small_world(...)
```

A.3 Step 2: Rewiring process

step2.m

```
function [mat, vec, dominant_freq, most_freq] = ...
    step2(t_end,phi,alpha,mat,vec,p,threshold)
%%%%%%%%%%%%%%%%%%%%%%%% step2: Runing the Simulation %%%%%%%%%%%%%%%%%%%
%%% t_end: number of iterations
%%% phi:  network reorganization rate
%%% alpha: innovation rate
%%% mat: initial connectivity matrix
%%% vec: initial idea vector
%%% p: initial number of opinions
%%% then it outputs:
%%% mat: connectivity matrix after simulation
%%% vec: idea vector after simulation
%%% dominant_ferq: the vector holding the frequency of dominant idea
%%% most_freq: the vector holding the index of dominating idea in each time

most_freq=zeros(1,t_end); %%% Vector for storing the index of the ...
    dominating idea in each time step.
dominant_freq=zeros(1,t_end); %%% Vector for storing the frequency of ...
    dominant idea in each time step.
a=size(mat);
n=a(1); %%% number of agents

for t=1:t_end
    x1=ceil(rand()*n); %%% choosing one person randomly for network ...
        reorganization or changing idea
    a1=rand();
    b1=phi;
    if a1<b1 %%% i.e with probability phi to reorganize the network
       v00=find(mat(x1,:)==1); %%% defining neighbours of x1
       v=find(vec(v00)≠vec(x1)); %%% define neighbours of x1 that do not ...
           have the same idea as x1
       if ¬isempty(v)
          x2=v(ceil(rand()*length(v))); %%% choosing one neighbour with ...
              different idea randomly to remove connection with
          mat(x1,x2)=0; %%% deletion of the edge between x1 and x2
          mat(x2,x1)=0; %%% deletion of the edge between x2 and x1
          similar_idea=find(vec==vec(x1)); %%% define the agents with ...
              the same idea as x1
          non_neighbor_sim_idea=setdiff(similar_idea,v00); %%% the ...
              agents with similar idea as x1 who are not neighbor of x1
          if ¬isempty(non_neighbor_sim_idea)
              x3=non_neighbor_sim_idea(ceil(rand()*length(non_neighbor_sim_idea))); ...
                  %%% choose x3 randomly among the agents with the same ...
                  idea and non-neighbor with x1 as the newly connected ...
                  agent to x
              mat(x1,x3)=1; %%% formation of new edge between x1 and x3
              mat(x3,x1)=1; %%% formation of new edge between x3 and x1
          else %%% we'll forget about network reorganization in this ...
              time by reforming the deleted edges.
              mat(x1,x2)=1;
              mat(x2,x1)=1;
          end
       end

    else  %%% otherwise change the idea of x1 to that of one of it's ...
        randomly chosen neighbours
       v2=find(mat(x1,:)==1); %%% defining the neighbours of x1
       if ¬isempty(v2)
          vv=vec(v2); %%% the corresponding ideas of the neighbours
          vvv=unique(vv); %%% vector of all the distinct ideas
          freq=zeros(1,length(vvv)); %%% vector for frequencies of the ideas
          for i=1:length(vvv) %%% to test for all distinct ideas
             if length(find(vv==vvv(i)))>(threshold*length(v2)) ...
                 %%%whether the frequency is larger than the threshold ...
                 [To include complex contagion definition]
                freq(i)=1;
             end
          end
          candidates=vvv(find(freq==1)); %%% The ideas meeting the threshold
          candidates_size=length(candidates); %%% The number of ideas ...
              meeting the threshold
          if candidates_size>0
             chosen=ceil(rand()*candidates_size);%%% Randomly choose one ...
                 of the candidates
             vec(x1)=candidates(chosen);  %%% change the idea of x1 to ...
                 the chosen idea
          end
       end
    end

    y=ceil(rand()*n); %%%choosing one person randomly for coming up with ...
        a new idea
    a2=rand();
    b2=alpha;
    if a2<b2 %%% i.e. with probability alpha to generate a novel idea
        bound=10^6; %%% to limit the index of new ideas to 10^6 which ...
            simulates nearly boundryless pool of ideas
        new_idea=ceil(rand()*bound)+p; %%% the index of new idea
        vec(y)=new_idea; %%% changing the idea of agent y to the novel one
    end
    %%%%
    most_freq(t)=mode(vec); %%% the dominant idea at time t
    dominant_freq(t)=length(find(vec==most_freq(t))); %%% the frequency ...
        of the dominant idea at time t
end
```

```
end
```

A.4 Step 3: Results

step3a.m

```
function n_index = step3a(mat,vec)
%%%%%%% step3a: defining the neighbourhood index between similar ideas ...
    %%%%%%%
%%% mat: adjacency matrix
%%% vec: idea vector
%%% it outputs: n_index which is the average neighbour index of the network
u=unique(vec); %%% collection of distinct ideas
w=zeros(1,length(u)); %%% initializing the vector for storing the ...
    neighbourhood index for each distinct idea
s=zeros(1,length(u)); %%% initializing the vector for storing the number ...
    of agents holding each distinct ideas
for i=1:length(u)
    x=find(vec==u(i));  %%% defining the set of agents holding the idea u(i)
    s(i)=length(x);
    if s(i)>1
    sum1=0; %%% number of agents with idea i which are in direct ...
        neighbourhood
    for k=1:(s(i)-1) %%% These two for loops are used to test the ...
        neighborhoods of all distinct pairs
        for j=(k+1):s(i)
            if mat(x(k),x(j))==1 %%% to check if they are neighbours
                sum1=sum1+1;
            end
        end
    end
    max_neighbors=(s(i)*(s(i)-1)/2); %%% normalizing factor (i.e., the ...
        maximum number of pairs of distinct agents)
    w(i)=sum1/max_neighbors; %%% neighborhood index for idea 'i'
    else
        w(i)=0;
    end
end
norml=sum(s); %%% Normalizing factor
count=sum(w.*s); %%% The weighted sum of neighborhood index
n_index=count/norml; %%% average neighbour index of the whole network
end
```

steb3b.m

```
function intra_idea_distance = step3b(mat,vec,s,c)
%%%%%%%%%%%% step3b: Defining the average of the average shortest ...
    distance between agents holding the same idea %%%%%%%%%%%%%%
%%% mat: adjacency matrix
%%% vec: idea vector
%%% s: number of connected components
%%% c:  vector which assigns each node to a connected component
%%% and it outputs the average of average intra_distance between agents
%%% holding the same idea %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
unq=unique(vec); %%% collection of distinct ideas
mean_sum=zeros(1,length(unq)); %%% initializing the vector for storing ...
    the average distance between agents holding a particular idea
n=zeros(1,length(unq)); %%% initializing the corresponding vector for ...
    storing the number of agents holding a certain idea
mats=sparse(mat);

for i=1:length(unq)
    x=find(vec==unq(i));  %%% defining the set of agents holding the ...
        idea unq(i)
    n(i)=length(x); %%% number of agents holding the idea unq(i)
    nn=zeros(1,s); %%% initializing the vector which will store the ...
        number of agents holding the idea unq(i) who belong to the m-th ...
        connected component
    mean_sum0=zeros(1,s);%%% initializing the vector which stores the ...
        mean of the distances between agents holding the idea unq(i) who ...
        belong to each distinct connected component
    for m=1:s %%% for each connected component
        y=x(find(c(x)==m)); %%% the set of agents holding the idea ...
            unq(i) who belong to the m-th connected component
        nn(m)=length(y); %%% number of agents holding the idea unq(i) ...
            who belong to the m-th connected component
        sum0=0;
        if nn(m)>1
           for j=1:(nn(m)-1)
               for k=(j+1):nn(m)
                   dis=graphshortestpath(mats,y(j),y(k)); %%% distance ...
                       of each agent with similar idea from agent j ...
                       inside the group [needs the bioinformatics ...
                       toolbox to be installed]
                   sum0=sum0+dis; %%% sum of the distances between ...
                       agents holding the idea unq(i) who belong to m-th ...
                       connected component
               end
           end
           norml0=nn(m)*(nn(m)-1)/2; %%% normalizing factor
           mean_sum0(m)=sum0/norml0; %%% The avarage intra_idea distance ...
                between agents holding the idea unq(i) who belong to the ...
```

```
                 m-th connected component
         else
            mean_sum0(m)=0;
         end
     end

     if n(i)>1
        mean_sum(i)=(sum(mean_sum0.*nn))/(sum(nn)); %%% The avarage ...
            intra_idea distance  between agents holding the idea unq(i)
     else
        mean_sum(i)=0;
     end
end

intra_idea_distance=(sum(mean_sum.*n))/(sum(n)); %%% The avarage ...
    intra_idea distance for the whole network
end
```

step3e.m

```
function [ average_dominance_time ] = step3e( most_freq )
%%%%% this function calculates the average of dominance time for the ...
    dominating idea during the simulation
%%%%% most_freq: is the vector obtained from simulation which holds the ...
    index of dominating ideas for each time steps of the simulation
%%%%% it outputs the average_time which is the average of the dominance ...
    time for different dominance periods
temp=zeros(1,length(most_freq)); %%% a temporary array which will hold ...
    the dominating period for dominating idea
count=1; %%% is the number of consecutive time steps in which an idea is ...
    considered as dominating
ind=0; %%% is the index of domination period
for i=2:length(most_freq)
    if most_freq(i)==most_freq(i-1)
        count=count+1; %%% the number of consecutive steps is conted
    else
        ind=ind+1; %%% as soon as another domination period gets tarted ...
            the index of domination period adds by one
        temp(ind)=count; %%% the number of consecutive steps will be ...
            stored in the ind-th index of tmp
        count=0; %%% and count will be reset to zero to count the ...
            duration of the new domination period.
    end
end

if ind==0 %%% in case during the simulation, just one special idea ...
    remains dominating forever
    ind=ind+1;
    temp(ind)=count;
end

%%% from ind-th element till end the temp vector will remain zero we ...
    define a new vector as follows to store the corresponding non-zero ...
    elements of temp
final=zeros(1,ind);
for j=1:ind
    final(j)=temp(j);
end
average_dominance_time=mean(final);
end
```

step4a.m

```

function clust_coeff = step4a( mat )
% This function calculates the clustering coefficient of a network with
% Corresponding connectivity matrix: mat
% The approach used for calculation of clustering coeffient was the one ...
    which was proposed by Watts and Strogatz
% in which the clustering coeffient of the whole network equals to the
% average of the local clustering coeefient of all nodes:
% D. J. Watts and Steven Strogatz (June 1998). "Collective dynamics of ...
    'small-world' networks". Nature 393 (6684): 440 4 4 2 .
a=size(mat);
local_clust_coeff=zeros(1,a(1)); %%% the vector to store the local ...
    clustering coeffient of each node of the network
for i=1:a(1) %%% for each agent in the network
    x=find(mat(i,:)==1); %%% find all the neighbours of agent i
    count=0;
    l=length(x);
    if l>1
    for j=1:(l-1) %%% loop to investigate the neighbour relationship for ...
        all possible pairs among the neighbors of node i
        for k=(j+1):l
            if mat(x(j),x(k))==1
                count=count+1; % to count the number of neighbour ...
                    relationships
            end
        end
    end
    norml=l*(l-1)/2; %% normalizing factor (i.e. the number of distinct ...
        pairs)
    local_clust_coeff(i)=count/norml; %% local clustering coefficient of ...
        node i
    else
    local_clust_coeff(i)=0;
    end
end
clust_coeff=sum(local_clust_coeff)/a(1); %% the average clustering ...
    coefficient of entire network
end
```

step4b.m

```
function [dgr,frq] = step4b(mat)
%%% this function outputs the degree vector (dgr) and its corresponding
%%% frequency vector (frq) from the connectivity matrix (mat). In other ...
    words,
%%% first the degree of all nodes in the network will be calculated and
%%% then the set of unique degrees will be stored in the dgr vector and the
%%% corresponding frequency will be stored in vector frq. For example,
%%% degree x will be stored in the i-th element of the dgr vector, then ...
    the number of nodes whose degree equals x
%%% will be calculated and will be stored in the i-th element of vector frq.
a=size(mat);
degr=zeros(1,a(1)); %%% the vector which stores the degree of each node ...
    of the network
for i=1:a(1)
    degr(i)=sum(mat(i,:)); %%% degree for each node
end
dgr=unique(degr); %%% set of unique degrees of the nodes of the network
frq=zeros(1,length(dgr)); %%% the corresponding frequency vector
for j=1:length(dgr)
    frq(j)=length(find(degr==dgr(j))); %%% frq(j) is the number of nodes ...
        in the network whose degree equals to dgr(j)
end

end
```

step4d.m

```
function average_path_length = step4d( mat,s,c )
%This function outputs the average path length for the graph
%mat: connectivity matrix
%s: number of connected components
%c: the connected component vector
av_p_l=zeros(1,s); %%% the vector which will store the average path ...
    length for each connected component of the 'mat' network
n=zeros(1,s); %%%  the vector which will store the number of nodes ...
    belonging to each connected component.
sp=sparse(mat);

for i=1:s %%% for each connected component
    x=find(c==i); %%% charachterizing the nodes which belong to the i-th ...
        connected component
    a=length(x); %%% the number of nodes in the i-th connected component
    n(i)=a;
    count=0;
    if a>1
    for k=1:(a-1)
        for j=(k+1):a
        count=count+graphshortestpath(sp,x(k),x(j)); %%% sum of the path ...
            length between all pairs of node
        end
    end
    norml=a*(a-1)/2; %%% normalization factor
    av_p_l(i)=count/norml; %%% average of path length
    else
    av_p_l(i)=0;
    end
end
average_path_length=sum(av_p_l.*n)/sum(n); %%% weighted average path length
end
```

step4e.m

```
function graph_diameter = step4e( mat,s,c )
%This function outputs the diameter of the graph 'mat'
%mat: connectivity matrix
%s: number of connected components
%c: the connected component vector
sp=sparse(mat);
max_path=zeros(1,s);  %%% vector storing diameter for each connected ...
    components.
for i=1:s %%% for each connected component
    x=find(c==i); %%% charachterizing the nodes which belong to the i-th ...
        connected component
    a=length(x); %%% the number of nodes in the i-th connected component
    path_length=zeros(1,(a*(a-1)/2)); %%% The vector storing the ...
        shortest path between each pairs of nodes belonging to the ...
        connected component i
    count=0;
    if a>1
    for k=1:(a-1)
        for j=(k+1):a
            count=count+1;
            path_length(count)=graphshortestpath(sp,x(k),x(j)); %%% the ...
                vector storing the shortest path between pairs k and j
        end
    end
    max_path(i)=max(path_length); %% the maximum shortest path for ...
        connected component i
    else
    max_path(i)=0;
    end
end
graph_diameter=max(max_path); %% the maximum shortest path for the hole ...
    network (Diameter of the network)
end
```

A.5 Plote codes

plots_phase1.m

```
%%%%%%%%%%%%%%%%%% Plots of phase 1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%% Extracting information from the saved Mat files %%%%%%%
neighbor_ind=zeros(1,108);
int_id_dis=zeros(1,108);
nov_ind=zeros(1,108);
av_dom_tim=zeros(1,108);
dom_freq=zeros(108,1000);
alpha=[0.01,0.05,0.1];
phi=[0.1,0.3,0.5];
threshold=[0.001,0.01,0.05];
count=0;
 f=figure();
for i=1:4
    switch i
        case 1
        s1='Caveman';
        case 2
        s1='Random';
        case 3
        s1='Scale_free';
        case 4
        s1='Small_world';
    end
    for j=1:3
        s2=int2str(j);
        for k=1:3
            s3=int2str(k);
            for l=1:3
                count=count+1;
                s4=int2str(l);
                name=['phase1_',s1,'_',s2,'_',s3,'_',s4];
                a=load(name);
                neighbor_ind(count)=a.neighbor_index;
                int_id_dis(count)=a.intra_idea_distance;
                nov_ind(count)=a.nov_index;
                av_dom_tim(count)=a.average_dominance_time;
                dom_freq(count,:)=a.dominant_freq;
            end
        end
    end
end

%%% phase diagrams for each parameter pairs for the results of the effects
%%% of network structure on idea distribution.
```

```

for i=1:4
    switch i
        case 1
        s1='Caveman';
        val=0;
        case 2
        s1='Random';
        val=27;
        case 3
        s1='Scale free';
        val=54;
        case 4
        s1='Small world';
        val=81;
    end
    for j=1:4
        switch j
        case 1
        s2='neighbor index';
        main_vec=neighbor_ind;
        case 2
        s2='intra idea distance';
        main_vec=int_id_dis;
        case 3
        s2='nov index';
        main_vec=nov_ind;
        case 4
        s2='average dominance time';
        main_vec=av_dom_tim;
        end
        for k=1:3
             switch k
             case 1
             s3='(alpha:phi)';
             x=0:0.5:1;
             xl='alpha';
             y=0:0.5:1;
             yl='phi';
             for l=1:3
                 M=[main_vec(val+l),main_vec(val+l+9),main_vec(val+l+18);
                     main_vec(val+l+3),main_vec(val+l+12),main_vec(val+l+21
                 );main_vec(val+l+6),main_vec(val+l+15),main_vec(val+l+24)];
                 %
                 [xlab,ylab]=meshgrid(x,y);
                 hold on;
                 view(0,90);
                 surf(xlab,ylab,M,'EdgeColor','none');
                 colorbar;
                 set(gca,'FontSize',14)
```

```
            xlabel(xl);
            ylabel(yl);
            switch l
                case 1
                    s5='first';
                case 2
                    s5='second';
                case 3
                    s5='third';
            end
            name2=['phase diagram of ',s2,' for alpha versus phi','
                and','the', s5,' threshold','obtained for',s1,'network
                structure'];
            title(name2);
            name=['plotting1_','Pdiag_',s1,'_',s2,'_',s3,'_',s4,
                'AND','the', s5,' threshold'];
            saveas(f,name);
            hold off
            %
        end
        case 2
        s3='(alpha:threshold)';
        x=0:0.5:1;
        xl='alpha';
        y=0:0.5:1;
        yl='threshold';
        for l=1:3
            M=[main_vec(val+9*(l-1)+1),main_vec(val+9*(l-1)+2),
                main_vec(val+9*(l-1)+3);main_vec(val+9*(l-1)+4),
                main_vec(val+9*(l-1)+5),main_vec(val+9*(l-1)+6);
                main_vec(val+9*(l-1)+7),main_vec(val+9*(l-1)+8),
                main_vec(val+9*(l-1)+9)];
            %
            [xlab,ylab]=meshgrid(x,y);
            hold on;
            view(0,90);
            surf(xlab,ylab,M,'EdgeColor','none');
            colorbar;
            set(gca,'FontSize',14)
            xlabel(xl);
            ylabel(yl);
             switch l
                case 1
                    s5='first';
                case 2
                    s5='second';
                case 3
                    s5='third';
            end
            name2=['phase diagram of ',s2,' for alpha versus threshold
```

```
                    ',' and','the', s5,' phi','obtained for',s1,'network
                    structure'];
                title(name2);
                name=['plotting1_','Pdiag_',s1,'_',s2,'_',s3,'_',s4,'
                    and','the', s5,' phi'];
                saveas(f,name);
                hold off
                %
            end

            case 3
            s3='(phi:threshold)';
            x=0:0.5:1;
            xl='phi';
            y=0:0.5:1;
            yl='threshold';
            for l=1:3
                M=[main_vec(val+3*(l-1)+1),main_vec(val+3*(l-1)+2),
                    main_vec(val+3*(l-1)+3);main_vec(val+3*(l-1)+10),
                    main_vec(val+3*(l-1)+11),main_vec(val+3*(l-1)+12);
                    main_vec(val+3*(l-1)+19),main_vec(val+3*(l-1)+20),
                    main_vec(val+3*(l-1)+21)];
                %
                [xlab,ylab]=meshgrid(x,y);
                hold on;
                view(0,90);
                surf(xlab,ylab,M,'EdgeColor','none');
                colorbar;
                set(gca,'FontSize',14)
                xlabel(xl);
                ylabel(yl);
                 switch l
                    case 1
                        s5='first';
                    case 2
                        s5='second';
                    case 3
                        s5='third';
                end
                name2=['phase diagram of ',s2,' for phi versus threshold'
                    ,' and','the', s5,' threshold','obtained for',s1,
                    'network structure'];
                title(name2);
                name=['plotting1_','Pdiag_',s1,'_',s2,'_',s3,'_',s4,
                    ' and','the', s5,' threshold'];
                saveas(f,name);
                hold off
                %
            end
            end
```

```
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Dependence of the property of each network to the parameters

for i=1:4
    switch i
        case 1
        s1='Caveman';
        val=0;
        case 2
        s1='Random';
        val=27;
        case 3
        s1='Scale free';
        val=54;
        case 4
        s1='Small world';
        val=81;
    end
    for j=1:4
        switch j
        case 1
        s2='neighborhood index';
        main_vec=neighbor_ind;
        case 2
        s2='intra idea distance';
        main_vec=int_id_dis;
        case 3
        s2='novelity index';
        main_vec=nov_ind;
        case 4
        s2='average dominance time';
        main_vec=av_dom_tim;
        end
        for k=1:3
            switch k
                case 1
                    s3='threshold';
                    f=figure();
                    for l=1:3
                        line=main_vec(3*(0:8)+val+l);
                        set(gca,'FontSize',14)
                        xlabel('Different pairs of alpha and phi values');
                        ylabel(s2);
                        namek=['Dependence of',' ',s2,' ','on',' ',s3,'
                            ','for',' ',s1,' ','network structure'];
                        title(namek);

                switch l
                    case 1
                         plot(line,'--ks','LineWidth',2,
                         'MarkerEdgeColor','k','MarkerFaceColor',
                         'k','MarkerSize',5)
                         hold on
                    case 2
                         plot(line,'--bs','LineWidth',2,
                         'MarkerEdgeColor','b','MarkerFaceColor',
                         'b','MarkerSize',5)
                         hold on
                    case 3
                         plot(line,'--rs','LineWidth',2,
                         'MarkerEdgeColor','r','MarkerFaceColor',
                         'r','MarkerSize',5)
                end
                legend('threshold=0.001','threshold=0.01',
                'threshold=0.05')
                name=['plotting1_',s1,'_',s2,'_',
                    'threshold_sensitivity'];
                saveas(f,name);
            end

        case 2
            s3='alpha';
            f=figure();
            for l=1:3
                line=main_vec([1,2,3,10,11,12,19,20,21]+val+3*(l-1));
                set(gca,'FontSize',14)
                xlabel('Different pairs of threshold and phi ...
                   values');
                ylabel(s2);
                namek=['Dependence of',' ',s2,' ','on',' ',s3,' ',
                    'for',' ',s1,' ','network structure'];
                title(namek);

                switch l
                    case 1
                         plot(line,'--ks','LineWidth',2,
                         'MarkerEdgeColor','k','MarkerFaceColor',
                         'k','MarkerSize',5)
                         hold on
                    case 2
                         plot(line,'--bs','LineWidth',2,
                         'MarkerEdgeColor','b','MarkerFaceColor',
                         'b','MarkerSize',5)
                         hold on
                    case 3
                         plot(line,'--rs','LineWidth',2,
                                    'MarkerEdgeColor','r','MarkerFaceColor',
                                    'r','MarkerSize',5)
                        end
                        legend('alpha=0.01','alpha=0.05','alpha=0.10')
                        name=['plotting1_',s1,'_',s2,'_',
                            'alpha_sensitivity'];
                        saveas(f,name);
                    end

                case 3
                    s3='phi';
                    f=figure();
                    for l=1:3
                        line=main_vec((1:9)+val+9*(l-1));
                        set(gca,'FontSize',14)
                        xlabel('Different pairs of threshold and alpha ...
                            values');
                        ylabel(s2);
                         namek=['Dependence of',' ',s2,' ','on',' ',s3,' ',
                             'for',' ',s1,' ','network structure'];
                        title(namek);

                        switch l
                            case 1
                                 plot(line,'--ks','LineWidth',2,
                                 'MarkerEdgeColor','k',
                                 'MarkerFaceColor','k','MarkerSize',5)
                                 hold on
                            case 2
                                 plot(line,'--bs','LineWidth',2,
                                 'MarkerEdgeColor','b','MarkerFaceColor',
                                 'b','MarkerSize',5)
                                 hold on
                            case 3
                                 plot(line,'--rs','LineWidth',2,
                                 'MarkerEdgeColor','r','MarkerFaceColor',
                                 'r','MarkerSize',5)
                        end
                        legend('phi=0.1','phi=0.3','phi=0.5')
                        name=['plotting1_',s1,'_',s2,'_','phi_sensitivity'];
                        saveas(f,name);
                    end
            end
        end
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Plotting each network feature for each set of
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% parameters %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:4
```

```
    switch i
        case 1
            f=figure();
            line1=neighbor_ind(1:27);
            line2=neighbor_ind(28:54);
            line3=neighbor_ind(55:81);
            line4=neighbor_ind(82:108);
            plot(line1,'--ks','LineWidth',2,'MarkerEdgeColor','k',
            'MarkerFaceColor','k','MarkerSize',5)
            hold on
            plot(line2,'--gs','LineWidth',2,'MarkerEdgeColor','g',
            'MarkerFaceColor','g','MarkerSize',5)
            hold on
            plot(line3,'--bs','LineWidth',2,'MarkerEdgeColor','b',
            'MarkerFaceColor','b','MarkerSize',5)
            hold on
            plot(line4,'--rs','LineWidth',2,'MarkerEdgeColor','r',
            'MarkerFaceColor','r','MarkerSize',5)
            set(gca,'FontSize',14)
            xlabel('Different triplets of alpha ,phi and threshold values')
            ylabel('Neighborhood index')
            title('The influence of network structure on neighborhood
            index');
            legend('Caveman Network','Random Network','Scale Free Network',
            'Small World Network')
            name=('Plotting1_Network Comparison_Neighborhood_index');
            saveas(f,name);
        case 2
            f=figure();
            line1=int_id_dis(1:27);
            line2=int_id_dis(28:54);
            line3=int_id_dis(55:81);
            line4=int_id_dis(82:108);
            plot(line1,'--ks','LineWidth',2,'MarkerEdgeColor','k',
            'MarkerFaceColor','k','MarkerSize',5)
            hold on
            plot(line2,'--gs','LineWidth',2,'MarkerEdgeColor','g',
            'MarkerFaceColor','g','MarkerSize',5)
            hold on
            plot(line3,'--bs','LineWidth',2,'MarkerEdgeColor','b',
            'MarkerFaceColor','b','MarkerSize',5)
            hold on
            plot(line4,'--rs','LineWidth',2,'MarkerEdgeColor','r',
            'MarkerFaceColor','r','MarkerSize',5)
            set(gca,'FontSize',14)
            xlabel('Different triplets of alpha ,phi and threshold values')
            ylabel('Intra-idea distance')
            title('The influence of network structure on intra-idea ...
                distance')
            legend('Caveman Network','Random Network','Scale Free Network',
            'Small World Network')
            name=('Plotting1_Network Comparison_intra_idea_distance');
            saveas(f,name);
        case 3
            f=figure();
            line1=nov_ind(1:27);
            line2=nov_ind(28:54);
            line3=nov_ind(55:81);
            line4=nov_ind(82:108);
            plot(line1,'--ks','LineWidth',2,'MarkerEdgeColor','k',
            'MarkerFaceColor','k','MarkerSize',5)
            hold on
            plot(line2,'--gs','LineWidth',2,'MarkerEdgeColor','g',
            'MarkerFaceColor','g','MarkerSize',5)
            hold on
            plot(line3,'--bs','LineWidth',2,'MarkerEdgeColor','b',
            'MarkerFaceColor','b','MarkerSize',5)
            hold on
            plot(line4,'--rs','LineWidth',2,'MarkerEdgeColor','r',
            'MarkerFaceColor','r','MarkerSize',5)
            set(gca,'FontSize',14)
            xlabel('Different triplets of alpha ,phi and threshold values')
            ylabel('Novelity Index')
            title('The influence of network structure on Novelity index')
            legend('Caveman Network','Random Network','Scale Free Network',
            'Small World Network')
            name=('Plotting1_Network Comparison_intra_novelity_index');
            saveas(f,name);
        case 4
            f=figure();
            line1=av_dom_tim(1:27);
            line2=av_dom_tim(28:54);
            line3=av_dom_tim(55:81);
            line4=av_dom_tim(82:108);
            plot(line1,'--ks','LineWidth',2,'MarkerEdgeColor','k',
            'MarkerFaceColor','k','MarkerSize',5)
            hold on
            plot(line2,'--gs','LineWidth',2,'MarkerEdgeColor','g',
            'MarkerFaceColor','g','MarkerSize',5)
            hold on
            plot(line3,'--bs','LineWidth',2,'MarkerEdgeColor','b',
            'MarkerFaceColor','b','MarkerSize',5)
            hold on
            plot(line4,'--rs','LineWidth',2,'MarkerEdgeColor','r',
            'MarkerFaceColor','r','MarkerSize',5)
            set(gca,'FontSize',14)
            xlabel('Different triplets of alpha ,phi and threshold values')
            ylabel('Average Dominance Time')
            title('The influence of network structure on Average Dominance
            Time')
            legend('Caveman Network','Random Network','Scale Free Network',
            'Small World Network')
            name=('Plotting1_Network Comparison_Average Dominance Time');
            saveas(f,name);
    end
end


%%%%%%%%%%%%%%%%%%%% plotting frequency of dominant idea over time for each
%%%%%%%%%%%%%%%%%%%% parameter sets %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
count=0;
for choice1=1:3
    switch choice1
        case 1
            s1='phi=0.1';
            m1='1';
        case 2
            s1='phi=0.3';
            m1='2';
        case 3
            s1='phi=0.5';
            m1='3';
    end

    for choice2=1:3
        switch choice2
            case 1
                s2='alpha=0.01';
                m2='1';
            case 2
                s2='alpha=0.05';
                m2='2';
            case 3
                s2='alpha=0.1';
                m2='3';
        end

        for choice3=1:3
            switch choice3
                case 1
                    s3='threshold=0.001';
                    m3='1';
                case 2
                    s3='threshold=0.01';
                    m3='2';
                case 3
                    s3='threshold=0.05';
                    m3='3';
            end
            count=count+1;
```

```
              line1=dom_freq(count,:);
              line2=dom_freq(27+count,:);
              line3=dom_freq(54+count,:);
              line4=dom_freq(81+count,:);
              f=figure();
              plot(line1,'--ks','LineWidth',2,'MarkerEdgeColor','k',
              'MarkerFaceColor','k','MarkerSize',5)
              hold on
              plot(line2,'--gs','LineWidth',2,'MarkerEdgeColor','g',
              'MarkerFaceColor','g','MarkerSize',5)
              hold on
              plot(line3,'--bs','LineWidth',2,'MarkerEdgeColor','b',
              'MarkerFaceColor','b','MarkerSize',5)
              hold on
              plot(line4,'--rs','LineWidth',2,'MarkerEdgeColor','r',
              'MarkerFaceColor','r','MarkerSize',5)
              set(gca,'FontSize',14)
              xlabel('time')
              ylabel('Frequency of the dominant idea')
              title(['The influence of network structure on the frequency
                  of the dominant idea',' ',s1,' ',s2,' ',s3]);
              legend('Caveman Network','Random Network','Scale Free Network'
              ,'Small World Network')
              name=(['Plotting1_Network Comparison_frequency_dominant_idea-
                  ',m1,'_',m2,'_',m3]);
              saveas(f,name);
          end
      end
  end
```

plots_phase2.m

```
%%%%%%%%%%%%%%%%%%% Plots of phase 2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%% Extracting information from the saved Mat files %%%%%%%
%%%%
clust_coefficient=zeros(1,81);
s=zeros(1,81);
average_path_length=zeros(1,81);
diam=zeros(1,81);
dgr=cell(1,81);
frq=cell(1,81);
alpha=[0.01,0.05,0.1];
phi=[0.1,0.3,0.5];
threshold=[0.001,0.01,0.05];

count=0;

for i=1:3
    switch i
        case 1
        s1='Random';
        case 2
        s1='Parallel';
        case 3
        s1='Antiparallel';
    end
    for j=1:3
        s2=int2str(j);
        for k=1:3
            s3=int2str(k);
            for l=1:3
                count=count+1;
                s4=int2str(l);
                name=['phase2_',s1,'_',s2,'_',s3,'_',s4];
                a=load(name);
                clust_coefficient(count)=a.clust_coefficient;
                s(count)=a.s;
                average_path_length(count)=a.average_path_length;
                diam(count)=a.diam;
                dgr{1,count}=a.dgr;
                frq{1,count}=a.frq;
            end
        end
    end
end

f=figure();

```

```
%%% phase diagrams for each parameter pairs for the results of the effects
%%% of idea distribution on network structure.

for i=1:3
    switch i
        case 1
        s1='Random';
        val=0;
        case 2
        s1='Parallel';
        val=27;
        case 3
        s1='Antiparallel';
        val=54;
    end
    for j=1:4
        switch j
        case 1
        s2='Clusteing Coefficient';
        main_vec=clust_coefficient;
        case 2
        s2='Number of connected components';
        main_vec=s;
        case 3
        s2='Average path length';
        main_vec=average_path_length;
        case 4
        s2='Network diameter';
        main_vec=diam;
        end
        for k=1:3
             switch k
             case 1
             s3='(alpha:phi)';
             x=0:0.5:1;
             xl='alpha';
             y=0:0.5:1;
             yl='phi';
             for l=1:3
                 M=[main_vec(val+l),main_vec(val+l+9),main_vec(val+l+18)
                     ;main_vec(val+l+3),main_vec(val+l+12),
                     main_vec(val+l+21)
                     ;main_vec(val+l+6),main_vec(val+l+15),
                     main_vec(val+l+24)];
                 %
                 [xlab,ylab]=meshgrid(x,y);
                 hold on;
                 view(0,90);
                 surf(xlab,ylab,M,'EdgeColor','none');
                 colorbar;
```

```
                    set(gca,'FontSize',14)
                    xlabel(xl);
                    ylabel(yl);
                    switch l
                        case 1
                            s5='first';
                        case 2
                            s5='second';
                        case 3
                            s5='third';
                    end
                    name2=['phase diagram of ',s2,' for alpha versus phi',
                        ' and','the', s5,' threshold','obtained for',s1,
                        'idea distribution'];
                    title(name2);
                    name=['Plotting2_','Pdiag_',s1,'_',s2,'_',s3,'_',s4,
                        'AND','the', s5,' threshold'];
                    saveas(f,name);
                    hold off
                    %
                end
                case 2
                s3='(alpha:threshold)';
                x=0:0.5:1;
                xl='alpha';
                y=0:0.5:1;
                yl='threshold';
                for l=1:3
                    M=[main_vec(val+9*(l−1)+1),main_vec(val+9*(l−1)+2),
                        main_vec(val+9*(l−1)+3);main_vec(val+9*(l−1)+4),
                        main_vec(val+9*(l−1)+5),main_vec(val+9*(l−1)+6);
                        main_vec(val+9*(l−1)+7),main_vec(val+9*(l−1)+8),
                        main_vec(val+9*(l−1)+9)];
                    %
                    [xlab,ylab]=meshgrid(x,y);
                    hold on;
                    view(0,90);
                    surf(xlab,ylab,M,'EdgeColor','none');
                    colorbar;
                    set(gca,'FontSize',14)
                    xlabel(xl);
                    ylabel(yl);
                     switch l
                        case 1
                            s5='first';
                        case 2
                            s5='second';
                        case 3
                            s5='third';
                    end
                    name2=['phase diagram of ',s2,' for alpha versus
                        threshold',
                        ' and','the', s5,' phi','obtained for',s1,
                        'idea distribution'];
                    title(name2);
                    name=['Plotting2_','Pdiag_',s1,'_',s2,'_',s3,'_',s4,
                        ' and','the', s5,' phi'];
                    saveas(f,name);
                    hold off
                    %
                end

                case 3
                s3='(phi:threshold)';
                x=0:0.5:1;
                xl='phi';
                y=0:0.5:1;
                yl='threshold';
                for l=1:3
                    M=[main_vec(val+3*(l-1)+1),main_vec(val+3*(l-1)+2),
                        main_vec(val+3*(l-1)+3);main_vec(val+3*(l-1)+10),
                        main_vec(val+3*(l-1)+11),main_vec(val+3*(l-1)+12);
                        main_vec(val+3*(l-1)+19),main_vec(val+3*(l-1)+20),
                        main_vec(val+3*(l-1)+21)];
                    %
                    [xlab,ylab]=meshgrid(x,y);
                    hold on;
                    view(0,90);
                    surf(xlab,ylab,M,'EdgeColor','none');
                    colorbar;
                    set(gca,'FontSize',14)
                    xlabel(xl);
                    ylabel(yl);
                     switch l
                        case 1
                            s5='first';
                        case 2
                            s5='second';
                        case 3
                            s5='third';
                    end
                    name2=['phase diagram of ',s2,' for phi versus threshold',
                        ' and','the', s5,' threshold','obtained for',s1,
                        'idea distribution'];
                    title(name2);
                    name=['Plotting2_','Pdiag_',s1,'_',s2,'_',s3,'_',s4,
                        ' and','the', s5,' threshold'];
                    saveas(f,name);
                    hold off
                    %
                end
                end
            end
        end
    end

    %%% Dependence of the property of each network to the parameters

    for i=1:3

        switch i
            case 1
            s1='Random';
            val=0;
            case 2
            s1='Parallel';
            val=27;
            case 3
            s1='Antiparallel';
            val=54;
        end

        for j=1:4
            switch j
            case 1
            s2='clust_coefficient';
            main_vec=clust_coefficient;
            case 2
            s2='s';
            main_vec=s;
            case 3
            s2='average_path_length';
            main_vec=average_path_length;
            case 4
            s2='diam';
            main_vec=diam;
            end
            for k=1:3
                switch k
                    case 1
                        s3='threshold';
                        f=figure();
                        for l=1:3
                            line=main_vec(3*(0:8)+val+l);
                            set(gca,'FontSize',14)
                            xlabel('Different pairs of alpha and phi values');
                            ylabel(s2);
                            namek=['Dependence of',' ',s2,' ','on',' ',s3,' ',
                                'for',' ',s1,' ','idea distribution'];
                            title(namek);

                switch l
                    case 1
                         plot(line,'--ks','LineWidth',2,
                         'MarkerEdgeColor','k','MarkerFaceColor',
                         'k','MarkerSize',5)
                         hold on
                    case 2
                         plot(line,'--bs','LineWidth',2,
                         'MarkerEdgeColor','b','MarkerFaceColor',
                         'b','MarkerSize',5)
                         hold on
                    case 3
                         plot(line,'--rs','LineWidth',2,
                         'MarkerEdgeColor','r','MarkerFaceColor',
                         'r','MarkerSize',5)
                end
                legend('threshold=0.001','threshold=0.01',
                'threshold=0.05')
                name=['plotting1_',s1,'_',s2,'_',
                    'threshold_sensitivity'];
                saveas(f,name);
            end

        case 2
            s3='alpha';
            f=figure();
            for l=1:3
                line=main_vec([1,2,3,10,11,12,19,20,21]+val+3*(l-1)
                );
                set(gca,'FontSize',14)
                xlabel('Different pairs of threshold and phi values
                ');
                ylabel(s2);
                namek=['Dependence of',' ',s2,' ','on',' ',s3,' ',
                    'for',' ',s1,' ','idea distribution'];
                title(namek);

                switch l
                    case 1
                         plot(line,'--ks','LineWidth',2,
                         'MarkerEdgeColor','k','MarkerFaceColor',
                         'k','MarkerSize',5)
                         hold on
                    case 2
                         plot(line,'--bs','LineWidth',2,
                         'MarkerEdgeColor','b','MarkerFaceColor',
                         'b','MarkerSize',5)
                         hold on
                    case 3
```

```
                                  plot(line,'--rs','LineWidth',2,
                                  'MarkerEdgeColor','r','MarkerFaceColor',
                                  'r','MarkerSize',5)
                         end
                         legend('alpha=0.01','alpha=0.05','alpha=0.10')
                         name=['plotting1_',s1,'_',s2,'_','alpha_sensitivity
                             '];
                         saveas(f,name);
                     end

                 case 3
                     s3='phi';
                     f=figure();
                     for l=1:3
                         line=main_vec((1:9)+val+9*(l-1));
                         set(gca,'FontSize',14)
                         xlabel('Different pairs of threshold and alpha
                         values');
                         ylabel(s2);
                          namek=['Dependence of',' ',s2,' ','on',' ',s3,'
                              ','for',' ',s1,' ','idea distribution'];
                         title(namek);

                         switch l
                             case 1
                                  plot(line,'--ks','LineWidth',2,
                                  'MarkerEdgeColor','k','MarkerFaceColor',
                                  'k','MarkerSize',5)
                                  hold on
                             case 2
                                  plot(line,'--bs','LineWidth',2,
                                  'MarkerEdgeColor','b','MarkerFaceColor',
                                  'b','MarkerSize',5)
                                  hold on
                             case 3
                                  plot(line,'--rs','LineWidth',2,
                                  'MarkerEdgeColor','r','MarkerFaceColor',
                                  'r','MarkerSize',5)
                         end
                         legend('phi=0.1','phi=0.3','phi=0.5')
                         name=['plotting1_',s1,'_',s2,'_','phi_sensitivity'];
                         saveas(f,name);
                     end
             end
         end
     end
end


%%%%%%%% generating plots for each features of network structure for each
```

```
%%%%%%%% parameter sets %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:4
    switch i
        case 1
            f=figure();
            line1=clust_coefficient(1:27);
            line2=clust_coefficient(28:54);
            line3=clust_coefficient(55:81);
            plot(line1,'--ks','LineWidth',2,'MarkerEdgeColor','k',
            'MarkerFaceColor','k','MarkerSize',5)
            hold on
            plot(line2,'--gs','LineWidth',2,'MarkerEdgeColor','g',
            'MarkerFaceColor','g','MarkerSize',5)
            hold on
            plot(line3,'--bs','LineWidth',2,'MarkerEdgeColor','b',
            'MarkerFaceColor','b','MarkerSize',5)
            set(gca,'FontSize',14)
            xlabel('Different triplets of alpha ,phi and threshold values')
            ylabel('Clustering Coefficient')
            title('The influence of idea distribution on clustering
            coefficient');
            legend('Random distribution','Parallel distribution',
            'Antiparallel distribution')
            name=('Plotting2_idea_Comparison_clustcoeff');
            saveas(f,name);
        case 2
            f=figure();
            line1=s(1:27);
            line2=s(28:54);
            line3=s(55:81);
            plot(line1,'--ks','LineWidth',2,'MarkerEdgeColor','k',
            'MarkerFaceColor','k','MarkerSize',5)
            hold on
            plot(line2,'--gs','LineWidth',2,'MarkerEdgeColor','g',
            'MarkerFaceColor','g','MarkerSize',5)
            hold on
            plot(line3,'--bs','LineWidth',2,'MarkerEdgeColor','b',
            'MarkerFaceColor','b','MarkerSize',5)
            set(gca,'FontSize',14)
            xlabel('Different triplets of alpha ,phi and threshold values')
            ylabel('Number of connected components')
            title('The influence of idea distribution on the number of
            connected components')
            legend('Random distribution','Parallel distribution',
            'Antiparallel distribution')
            name=('Plotting2_idea_Comparison_conncmp');
            saveas(f,name);
        case 3
            f=figure();
            line1=average_path_length(1:27);
            line2=average_path_length(28:54);
            line3=average_path_length(55:81);
            plot(line1,'--ks','LineWidth',2,'MarkerEdgeColor','k',
            'MarkerFaceColor','k','MarkerSize',5)
            hold on
            plot(line2,'--gs','LineWidth',2,'MarkerEdgeColor','g',
            'MarkerFaceColor','g','MarkerSize',5)
            hold on
            plot(line3,'--bs','LineWidth',2,'MarkerEdgeColor','b',
            'MarkerFaceColor','b','MarkerSize',5)
            set(gca,'FontSize',14)
            xlabel('Different triplets of alpha ,phi and threshold values')
            ylabel('average path length')
            title('The influence of idea distribution on average path ...
                length')
            legend('Random distribution','Parallel distribution',
            'Antiparallel distribution')
            name=('Plotting2_idea_Comparison_average_path_length');
            saveas(f,name);
        case 4
            f=figure();
            line1=diam(1:27);
            line2=diam(28:54);
            line3=diam(55:81);
            plot(line1,'--ks','LineWidth',2,'MarkerEdgeColor','k',
            'MarkerFaceColor','k','MarkerSize',5)
            hold on
            plot(line2,'--gs','LineWidth',2,'MarkerEdgeColor','g',
            'MarkerFaceColor','g','MarkerSize',5)
            hold on
            plot(line3,'--bs','LineWidth',2,'MarkerEdgeColor','b',
            'MarkerFaceColor','b','MarkerSize',5)
            set(gca,'FontSize',14)
            xlabel('Different triplets of alpha ,phi and threshold values')
            ylabel('Network Diameter')
            title('The influence of idea distribution on Network Diameter')
            legend('Random distribution','Parallel distribution',
            'Antiparallel distribution')
            name=('Plotting2_idea_Comparison_diam');
            saveas(f,name);
    end
end

%%%%%%%%% plots for degree distribution resulting from different idea
%%%%%%%%% distribution
count=0;
for choice1=1:3
    switch choice1
        case 1
```

```
                s1='phi=0.1';
                m1='1';
            case 2
                s1='phi=0.3';
                m1='2';
            case 3
                s1='phi=0.5';
                m1='3';
        end

        for choice2=1:3
            switch choice2
                case 1
                    s2='alpha=0.01';
                    m2='1';
                case 2
                    s2='alpha=0.05';
                    m2='2';
                case 3
                    s2='alpha=0.1';
                    m2='3';
            end

            for choice3=1:3
                switch choice3
                    case 1
                        s3='threshold=0.001';
                        m3='1';
                    case 2
                        s3='threshold=0.01';
                        m3='2';
                    case 3
                        s3='threshold=0.05';
                        m3='3';
                end
                count=count+1;
                line1=dgr{count};
                mine1=frq{count}/1000;
                line2=dgr{27+count};
                mine2=frq{27+count};
                line3=dgr{54+count};
                mine3=frq{54+count};
                f=figure();
                loglog(line1,mine1,'--ks','LineWidth',2,'MarkerEdgeColor','k',
                'MarkerFaceColor','k','MarkerSize',5)
                hold on
                plot(line2,mine2,'--rs','LineWidth',2,'MarkerEdgeColor','r',
                'MarkerFaceColor','r','MarkerSize',5)
                hold on
                plot(line3,mine3,'--bs','LineWidth',2,'MarkerEdgeColor','b',
                'MarkerFaceColor','b','MarkerSize',5)
                set(gca,'FontSize',14)
                xlabel('Log(degree)')
                ylabel('Log(frequency)')
                title(['The influence of idea distribution on the degree
                    distribution',' ',s1,' ',s2,' ',s3]);
                legend('Random distribution','Parallel distribution',
                'Antiparallel distribution')
                name=(['Plotting2_idea_comparison_degree_distribution-'
                    ,m1,'_',m2,'_',m3]);
                saveas(f,name);
            end
        end
    end
```

References

Barabási, A. and R. Albert (1999). Emergence of scaling in random networks. *science 286*(5439), 509–512.

Brugger, S. and C. Schwirzer (2011). Opinion formation by "employed agents" in social networks. Project work done in the course Lecture with Computer Exercises: Modelling and Simulating Social Systems with MATLAB, May 2011.

Centola, D. and M. Macy (2007). Complex contagions and the weakness of long ties. *American Journal of Sociology 113*(3), 702–734.

Erdős, P. and A. Rényi (1960). On the evolution of random graphs. *Magyar Tud. Akad. Mat. Kutató Int. Közl 5*, 17–61.

Holme, P. and M. Newman (2006). Nonequilibrium phase transition in the coevolution of networks and opinions. *Physical Review E 74*(5), 056108.

Watts, D. (2003). *Small worlds: the dynamics of networks between order and randomness.* Princeton university press.

Watts, D. J. and S. H. Strogatz (1998, June). Collective dynamics of 'small-world' networks. *Nature 393*(6684), 440–442.

Wikipedia (2011). Erdős-rényi model – wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/Erd%C5%91s%E2%80%93R%C3%A9nyi_model`.