



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

Rzgar Hosseini

Contents

1	Abstract	5
2	Individual contributions	6
3	Introduction and Motivation	7
3.1	Fundamental Questions	7
3.1.1	Effects of Network Structure on Idea Distribution	7
3.1.2	Effects of Idea Distribution on Network Structure	8
4	Description of the Model	10
4.1	Networks	10
4.1.1	Random Graph	11
4.1.2	Caveman Graph	11
4.1.3	Small World Graph	11
4.1.4	Scale-Free Graph	11
4.2	Ideas	12
5	Implementation	13
5.1	Step 1: Generating structure networks and initial idea distribution . .	13
5.2	Step 2: update	13
5.3	Step 3: Getting results	14
6	Simulation Results and Discussion	16
6.1	Effects of Network Structure on Idea Distribution	16
6.1.1	Intra-Idea Distance and Neighbourhood Index	17
6.1.2	Frequency of Dominance	18
6.2	Effects of Idea Distribution on Network Structure	19
6.2.1	Clustering Coefficient	19
6.2.2	Average Path Length	19
6.2.3	Network Diameter	20
6.2.4	Degree Distribution	20
6.3	Discussion	21
7	Summary and Outlook	23
A	MATLAB code	25
A.1	Main script	25
A.2	Step 1: generation of structure networks	29
A.3	Step 2: Rewiring process	39

A.4	Step 3: Results	41
A.5	Plotes code	47

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 DEFINITION OF TERMS****. 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.

3.1.1 Effects of Network Structure on Idea Distribution

Given a starting network and a random idea distribution, how do different network structures 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 look 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 *****TABLE DEFINITIONS OF TERMS*** for definitions), 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 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 (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 and the number of connected components will likely decrease, and the diameter and the average path length will decrease as well 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 characteristics 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 (≥ 1) or a fraction (between zero and one, inclusive). Our simulation used the second formulation since the degree of nodes varied across network structures.

4.1 Networks

For the purposes of our simulations, we used four network structures. *******NETWORK STRUCTURE FIGURE****** illustrates them. These structures can be characterized by properties such as average shortest path lengths, clustering coefficients, and the degree of connectivity (see *******TABLE DEFINITIONS OF TERMS****** 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 30). For further details about parameter values, see *******TABLE WITH PARAMETERS ******.

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 as defined by Barabási and Albert (1999) and implemented by Brugger and Schwirzer (2011).

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 ‘antiparallel’ 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. *****FIGURE OF IDEA DISTRIBUTION***** illustrates these idea distributions. 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 connections of nodes 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 to the opinion 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 in three steps.

Step 1: Structure network is chosen and we generate the adjacency matrix corresponding to that structure. Also is generated the initial idea distribution.

Step 2: The updating processes is done using the chosen structure network.

Step 3: At this step a series of functions are called to getting the results.

5.1 Step 1: Generating structure networks and initial idea distribution

For the first phase one of the functions **step1_scalefree**, **step1_caveman**, **step1_smallworld** and **step1_randomgraph** is called, depending in 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 is done, in the first phase we have a random initial distribution.

For the second phase we just generate the adjacency matrix for the caveman structure network, and then we chose among three different initial idea distributions: random, parallel or nonparallel. 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.

Rewiring simulation is implemented in the file **step2.m** and done by function **step2** which requires the next 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 dominant idea
- **most_freq**: the vector holding the index of dominating idea in each time

The function **step2** in which the updating is executed follows the next algorithm:

Algorithm 1 Update process

```

for each of the iterations do
  choose a person  $x1$  in the network at random
  generate a random number  $a1$ 
  if  $a1 < \phi$  then                                 $\triangleright$  With probability  $\phi$  we reorganize the network
    eliminate the connection between  $x1$  and one of his neighbors with different idea
    select another person at random among the persons with the same idea than  $x1$ 
    and that are not already neighbors of  $x1$  and create a connection between them
  else change the idea of  $x1$  to one of the ideas of his neighbors which meet the threshold
  end if
  choose a person  $y$  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     $\triangleright$  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 is necessary to search for parameters that reflects the final distribution of ideas. In this study we observe the following parameters:

- **n_index** which is the average neighbor index of the network
- **intra_idea_distance** which is the average of the average shortest distance between agents holding the same idea

- **average_dominance_time** which is the average of the dominance time for different dominance periods

All this parameters are obtained in **step3a.m**, **step3b.m** and **step 3d.m** which are called in **main_script.m**.

In phase 2 we want to find out how the distribution of ideas change the network structure, for this reason we just look at a fixed structure network, that would be the caveman structure. After step 2 we want to see how the initial structure was modified by the rewiring process, in order to see this we look for the next parameters

- **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). 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 OF DEFINITIONS***). 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 antiparallel to the structure) to a caveman network (see ***TABLE OF DEFINITIONS*** 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 1.1**). 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 1.2**). 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.

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 1.5-1.8**). 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 1.9-1.12**). 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 (**FIGURES 1.13-1.16**).

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 (**FIGURES 1.17-1.20**).

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 1.21-1.24**).

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 1.25**). 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 neighbours' 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.

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 1.3**). 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 1.4** 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 antiparallel 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 2.4****), the remaining features changed. The parameter α did not change these effects for any of the idea distributions (see ***FIGURES 2.6-2.14***). 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 2.3***). 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, whereas more rewiring would have occurred 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 antiparallel distributions (***FIGURES 2.21-2.23***). 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 (***FIGURES 2.30-2.32***).

6.2.2 Average Path Length

The average path length of resulting networks was larger when a parallel idea distribution was used (***FIGURE 2.1****). 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 pronounced for the random and anti-parallel idea distribution cases (**FIGURE 2.15-2.17**). This is another intuitive result since more rewiring naturally disturbs the rigid structure of a cave-man network, 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 (**FIGURES 2.24-2.26**). 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 antiparallel 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 2.2**). 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 occurred.

Dependence on Parameters As ϕ increased, the network diameter decreased for all idea distributions, and slightly less for the parallel distribution (**FIGURES 2.18-2.20**). 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 (**FIGURES 2.27-2.29**).

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 2.5**). 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 antiparallel 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.

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 antiparallel 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 (the intra-idea distance and neighbourhood index) varied more with the innovation rate α for less structured networks, and varied more with the complex contagion threshold δ for the most rigid structure - the caveman network. Parameter values (ϕ , δ , and α) seemed to vary the average dominance time more than the network structures did, 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 thresholds 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) do 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.

||||| HEAD =====

A MATLAB code

A.1 Main script

```
1 %% The first phase: Simulation to study the influence of network ...
   structure on the ...
   opinion%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %% The following parameters will remain constant during this study,
3 %% so we won't play with them %%
4 n=1000; %% the number of agents %%
5 m=40; %% initial number of clusters for caveman matrix%%
6 p=40; %% initial number of opinions%%
7 t_end=1000; %% number of iterations%%
8 %% for the following parameters we'll run different simulations based on
9 %% combinatorial complexity of the parameters%%
10 phi_choices=[0.1,0.3,0.5]; %% network reorganization rate%%
11 alpha_choices=[0.01,0.05,0.10]; %% innovation rate %%
12 threshold_choices=[0.001,0.01,0.05]; %% threshold for complex contagion %%
13
14
15 %% a totally random idea distribution , independent of the connectivity
16 %% matrix, so applicable for every network structure is defined for this
17 %% phase %%
18 vec1=zeros(1,n);
19 for i=1:n
20     vec1(i)=ceil(rand()*p);
21 end
22
23 for choice1=1:4
24 %% Step1: Definition of Different initial matrices ...
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
25 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% For each simulation, we must choose one of the ...
   following connectivity matrices %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
26 switch choice1
27     case 1
28         %% option 1: Caveman Connectivity Matrix
29         mat1=step1_caveman(n,m);
30         s1='Caveman';
31     case 2
32         %% option 2: Random Connectivity Matrix
33         prob=0.025; %% probability of edge formation between any pairs ...
   of edges
34         mat1=step1_randomgraph(n,prob);
35         s1='Random';
36     case 3
```

```

37     %% option 3: Scale Free Connectivity Matrix
38     m0=24; % number of initially placed nodes
39     m1=12; % number of nodes a new added node is connected to,  $1 \leq \dots$ 
40         m1 < m0
41     mat1=step1_scalefree(n, m0, m1);
42     s1='Scale-free';
43     case 4
44         %% option 4: Small world Connectivity Matrix
45         ka=24; %% mean degree (assumed to be an even integer)
46         beta=0.01; %% rewiring probability
47         mat1= step1_smallworld(n, ka, beta);
48         s1='Small-world';
49     end
50     %% Step2: Simulation ...
51     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
52     for choice2=1:3
53         phi=phi_choices(choice2);
54         for choice3=1:3
55             alpha=alpha_choices(choice3);
56             for choice4=1:3
57                 threshold=threshold_choices(choice4);
58                 [mat2,vec2,dominant_freq,most_freq]=step2(t_end,phi,alpha,mat1,vec1,p,threshold);
59                 %% obtaining the final matrix and vector after running ...
60                 simulation.
61             %%% We need step4c here, since it's outputs will be the input
62             %%% for step 3b
63             %%% step4c: number of connected components %%%
64             sp_mat2=sparse(mat2);
65             [s,c]=graphconncomp(sp_mat2); %% matlab built in function ...
66             'Bioinformatics Toolbox'
67             %% s: number of connected components
68             %% c: vector which assigns each node to a connected component
69
70             %% Step3: Results for structure to idea ...
71             %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
72             %%% step3a: defining the average intra_idea ...
73             neighbourhood_index %%%
74             neighbor_index=step3a(mat2,vec2);
75
76             %%% step3b: defining the average intra_idea distance ...
77             %%%
78             intra_idea_distance=step3b(mat2,vec2,s,c);
79
80             %%% step3c: frequency of dominant idea with respect to ...
81             time %%%

```

```

77         %% is the third output of the step2 function ...
           (dominant_freq)%%
78     dominant_freq;
79
80     % step3d: Fraction of novel ideas (novelity index) %
81     nov_index=(length(find(vec2>p)))/(length(vec2)); %% ...
           indicates the fraction of agents holding the newly ...
           generated ideas
82
83     % step3e: defining the average dominance time (the ...
           average amount of time in which the dominating idea ...
           keeps it's dominance over differnt dominance periods)
84     average_dominance_time=step3e(most_freq);
85
86     %% naming the file which saves the results
87     s2=int2str(choice2);
88     s3=int2str(choice3);
89     s4=int2str(choice4);
90     name=['phase1_',s1,'_',s2,'_',s3,'_',s4];
91     save(name);
92     end
93 end
94 end
95 end
96
97 clear;
98
99 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
100 %% The second phase: Simulation to study the influence of opinion on the ...
           network ...
           structure%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
101
102 %% The following three parameters will remain constant during this study,
103 %% so won't play with them %%
104 n=1200; %% the number of agents
105 m=40; %% initial number of clusters for caveman matrix
106 p=40; %% initial number of opinions
107 t_end=1000; %% number of iterations%%
108 %% for the following parameters we'll run different simulations based on
109 %% combinatorial complexity of the parameters%%
110 phi_choices=[0.1,0.3,0.5]; %% network reorganization rate%%
111 alpha_choices=[0.01,0.05,0.10]; %% innovation rate %%
112 threshold_choices=[0.001,0.01,0.05];%% threshold for complex contagion %%
113
114
115
116 %%%in this phase we'll keep connectivity matrix constant, so we only use
117 %%%Caveman connectivity matrix %%
118 mat1=step1_caveman(n,m);
119

```

```

120
121 for choice1=1:3
122 %% Step1: Definition of Different initial idea vectors ...
123     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
124     switch choice1
125     case 1
126         % option1: Random idea vector %
127         %% a totally random idea distribution , independent of the ...
128         connectivity
129         %% matrix, so applicable for every network structure
130         vec1=zeros(1,n);
131         for i=1:n
132             vec1(i)=ceil(rand()*p);
133         end
134         s1='Random';
135     case 2
136         % option2: Parallel idea vector %
137         %% This idea vector is applicable only for caveman ...
138         connectivity matrix in
139         %% which every agents inside a cluster have the same idea
140         vec1=zeros(1,n);
141         for i=1:(m-1) %% for each cluster
142             for j=1:ceil(n/m)
143                 vec1((i*ceil(n/m))+j)=i; %% all agents will hold the ...
144                 i-th idea
145             end
146         end
147         s1='Parallel';
148     case 3
149         % option3: Antiparallel idea vector %
150         %% This idea vector is applicable only for caveman ...
151         connectivity matrix in
152         %% which every agents inside a cluster have different idea
153         vec1=zeros(1,n);
154         for i=1:(m-1) %% for each cluster
155             for j=1:ceil(n/m)
156                 vec1((i*ceil(n/m))+j)=j; %% all agents will hold ...
157                 different idea
158             end
159         end
160         s1='Antiparallel';
161     end
162 %% Step2: Simulation ...
163     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
164 for choice2=1:3
165     phi=phi_choices(choice2);
166
167     for choice3=1:3
168         alpha=alpha_choices(choice3);
169     for choice4=1:3

```

```

163         threshold=threshold_choices(choice4);
164
165         [mat2,vec2,dominant_freq,most_freq]=step2(t_end,phi,alpha,mat1,vec1,p,threshold);
            %% obtaining the final matrix and vector after running ...
            simulation.
166
167
168         %% Step3: Results for idea to structure ...
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
169
170         % step4a: clustering coefficient of the final network ...
            %%%%%%%%%%
171         clust_coefficient=step4a(mat2);
172
173         % step4b: degree distribution of the final network ...
            %%%%%%%%%%
174         [dgr,frq]=step4b(mat2);
175         average_degree=sum(dgr.*frq)/sum(frq);
176
177         % step4c: number of connected components %%%%%%%%%%
178         sp_mat2=sparse(mat2);
179         [s,c]=graphconncomp(sp_mat2); %% matlab built in function ...
            'Bioinformatics Toolbox'
180         %% s: number of connected components
181         %% c: vector which assigns each node to a connected component
182
183         % step4d: average path length for the final network ...
            %%%%%%%%%%
184         average_path_length = step4d( mat2,s,c );
185
186         % step4e: Diameter of the network %%%%%%%%%%
187         diam=step4e(mat2,s,c);
188
189         %%naming and saving
190         s2=int2str(choice2);
191         s3=int2str(choice3);
192         s4=int2str(choice4);
193         name=['phase2_',s1,'-',s2,'-',s3,'-',s4];
194         save(name);
195     end
196 end
197 end
198 end
199
200 clear;

```

A.2 Step 1: generation of structure networks

randomgraph.m

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %%% code adopted from %%%
3 %%% Modeling and Simulating Social Systems with MATLAB %%%
4 %%% http://www.soms.ethz.ch/teaching/MatlabFall2012 %%%
5 %%% Authors: Stefan Brugger and Cristoph Schwirzer, 2011 %%%
6 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7
8 function final= stepl.randomgraph(n, p)
9 % Generates an undirected random graph (without self-loops) of size n (as
10 % described in the Erdoes-Renyi model)
11 %
12 % INPUT
13 % n: [1]: number of nodes
14 % p: [1]: probability that node i and node j, i != j, are connected by ...
    an edge
15 %
16 % OUTPUT
17 % final: [n n] full symmetric adjacency matrix representing the ...
    generated graph
18
19 % Note: A generation based on sprandsym(n, p) failed (for some values of p
20 % sprandsym was far off from the expected number of n*n*p non-zeros), ...
    therefore
21 % this longish implementation instead of just doing the following:
22 %
23 % B = sprandsym(n, p);
24 % A = (B-diag(diag(B))>0);
25 %
26
27 % Idea: first generate the number of non-zero values in every row for a ...
    general
28 % 0-1-adjacency matrix. For every row this number is distributed ...
    binomially with
29 % parameters n and p.
30 %
31 % The following lines calculate "rowSize = binoinv(rand(1, n), n, p)", ...
    just in a
32 % faster way for large values of n.
33
34 % generate a vector of n values chosen u.a.r. from (0,1)
35 v = rand(1, n);
36 % Sort them and calculate the binomial cumulative distribution function with
37 % parameters n and p at values 0 to n. Afterwards match the sorted random
38 % 0-1-values to those cdf-values, i.e. associate a binomial distributed ...
    value
39 % with each value in r. Each value in v also corresponds to a value in r:
40 % permute the values in rowSize s.t. they correspond to the order given ...
    in v.
41 [r index] = sort(v); % i.e. v(index) == r holds

```

```

42 rowSize = zeros(1, n);
43 j = 0;
44 binoCDF = cumsum(binopdf(0:n, n, p));
45 for i = 1:n
46     while j<n && binoCDF(j+1)<r(i)
47         j = j + 1;
48     end
49     rowSize(i) = j;
50 end
51 rowSize(index) = rowSize;
52
53 % for every row choose the non-zero entries in it
54 nNZ = sum(rowSize);
55 I = zeros(1, nNZ);
56 J = zeros(1, nNZ);
57 j = 1;
58 for i = 1:n
59     I(j:j+rowSize(i)-1) = i;
60     J(j:j+rowSize(i)-1) = randsample(n, rowSize(i));
61     j = j + rowSize(i);
62 end
63
64 % restrict I and J to indices that correspond to entries above the main ...
    diagonal
65 % and finally construct a symmetric sparse matrix using I and J
66 upperTriu = find(I<J);
67 I = I(upperTriu);
68 J = J(upperTriu);
69 A = sparse([I;J], [J;I], ones(1, 2*size(I, 2)), n, n);
70 final=full(A);
71 end % random_graph(...)

```

scalefree.m

```

1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %%% code adopted from %%%
3  %%% Modeling and Simulating Social Systems with MATLAB %%%
4  %%% http://www.soms.ethz.ch/teaching/MatlabFall2012 %%%
5  %%% Authors: Stefan Brugger and Cristoph Schwirzer, 2011 %%%
6  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7
8  function A = scalefree(n, m0, m)
9  % Use the Barabasi-Albert model to generate a scale free graph of size n (as
10 % described in Albert-Laszlo Barabasi & Reka Albert: "Emergence of scaling
11 % in random networks")
12 %
13 % INPUT
14 % n: [1]: number of nodes
15 % m0: [1]: number of initially placed nodes

```

```

16 % m: [1]: number of nodes a new added node is connected to,  $1 \leq m < m_0$ 
17 %
18 % OUPUT
19 % A: [n n] sparse symmetric adjacency matrix representing the generated ...
    graph
20
21 % Start with a graph of size m0 and add edges to this graph. Each of ...
    these m0
22 % nodes is connected to at least m nodes.
23 B = zeros(m0, m0);
24 for i = 1:m0
25     neighbors = randsample(m0-1, m);
26     neighbors = neighbors + (neighbors ≥ i);
27     B(i,neighbors) = 1;
28     B(neighbors,i) = 1;
29 end
30
31 % Create a vector of edges added so far, i.e. nodes edge(2*i) and ...
    edge(2*i-1),
32 %  $1 \leq i \leq nEdges$ , are connected by an edge.
33 [rows, columns] = find(triu(B));
34 nEdges = size(rows, 1);
35 edges = reshape([rows';columns'], 2*nEdges, 1);
36 edges = [edges; zeros(2*(n-m0)*m,1)];
37
38 % Add nodes m0+1:n, one at a time. Each node is connected to m existing ...
    nodes,
39 % where each of the existing nodes is chosen with a probability that is
40 % proportional to the number of nodes it is already connected to.
41 used = zeros(n, 1); % is a node already used in a timestep?
42 for i = m0+1:n
43     neighbors = zeros(1, m);
44     for j=1:m
45         k = edges(randi(2*nEdges));
46         while used(k)
47             k = edges(randi(2*nEdges));
48         end
49         used(k) = 1;
50         neighbors(j) = k;
51     end
52     used(neighbors) = 0;
53     edges(2*nEdges+1:2*nEdges+2*m) = reshape([repmat(i, 1, m); ...
        neighbors], ...
54         1, 2*m);
55     nEdges = nEdges+m;
56 end
57
58 % finally construct a symmetric adjacency matrix using the vector of edges
59 edges = edges(1:2*nEdges);
60 first = edges(1:2:end);

```



```

61 second = edges(2:2:end);
62 A = sparse([first;second], [second;first], ones(2*nEdges, 1), n, n);
63
64 end % scale_free(...)

```

smallworld.m

```

1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %%% code adopted from %%%
3  %%% Modeling and Simulating Social Systems with MATLAB %%%
4  %%% http://www.soms.ethz.ch/teaching/MatlabFall2012 %%%
5  %%% Authors: Stefan Brugger and Cristoph Schwirzer, 2011 %%%
6  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7
8  function A = smallworld(n, k, beta)
9  % Generate a small world graph using the "Watts and Strogatz model" as
10 % described in Watts, D.J.; Strogatz, S.H.: "Collective dynamics of
11 % 'small-world' networks."
12 % A graph with n*k/2 edges is constructed, i.e. the nodal degree is n*k for
13 % every node.
14 %
15 % INPUT
16 % n: [1]: number of nodes of the graph to be generated
17 % k: [1]: mean degree (assumed to be an even integer)
18 % beta: [1]: rewiring probability
19 %
20 % OUTPUT
21 % A: [n n] sparse symmetric adjacency matrix representing the generated ...
    graph
22
23 % Construct a regular lattice: a graph with n nodes, each connected to k
24 % neighbors, k/2 on each side.
25 kHalf = k/2;
26 rows = reshape(repmat([1:n]', 1, k), n*k, 1);
27 columns = rows+reshape(repmat([1:kHalf] [n-kHalf:n-1]), n, 1), n*k, 1);
28 columns = mod(columns-1, n) + 1;
29 B = sparse(rows, columns, ones(n*k, 1));
30 A = sparse([], [], [], n, n);
31
32 % With probability beta rewire an edge avoiding loops and link duplication.
33 % Until step i, only the columns 1:i are generated making implicit use ...
    of A's
34 % symmetry.
35 for i = [1:n]
36     % The i-th column is stored full for fast access inside the ...
        following loop.
37     col= [full(A(i, 1:i-1))'; full(B(i:end, i))];
38     for j = i+find(col(i+1:end))'
39         if (rand()<beta)

```

```

40         col(j)=0;
41         k = randi(n);
42         while k==i || col(k)==1
43             k = randi(n);
44         end
45         col(k) = 1;
46     end
47 end
48 A(:,i) = col;
49 end
50
51 % A is not yet symmetric: to speed things up, an edge connecting i and ...
52 %   j, i < j
53 % implies A(i,j)==1, A(i,j) might be zero.
54 T = triu(A);
55 A = T+T';
56 end % small_world(...)

```

step1_caveman.m

```

1 function cave_mat = step1_caveman(n,m)
2 %%%%%%%%%%% This function outputs a Caveman Matrix %%%%%%%%%%%
3 %%% n: the number of agents
4 %%% m: initial number of clusters
5 %%% p: initial maximum index of opinions
6 cave_mat=zeros(n,n); %Caveman Matrix
7
8 for i=1:n
9     x=ceil(i/(n/m));
10    for j=1:n
11        y=ceil(j/(n/m));
12        if x==y
13            if i≠j
14                cave_mat(i,j)=1; %definition of intracluster edges
15            end
16        end
17    end
18 end
19 %%% x1 and x2 for each cluster represent the two agents who interact with
20 %%% nearby clusters%%
21 x1=zeros(1,m);
22 x2=zeros(1,m);
23 for i=1:m
24     x1(i)=ceil(rand()*(n/m)+(n/m)*(i-1));
25     ind=0;
26     while(ind==0) % This loop is used to prevent x1 and x2 to make the ...
27         % same numbers
28         x2(i)=ceil(rand()*(n/m)+(n/m)*(i-1));

```

```

28         if x2(i)≠x1(i)
29             ind=1;
30         end
31     end
32 end
33 %%% definition of intercluster edges
34 for i=1:(m-1)
35     cave_mat(x2(i),x1(i+1))=1;
36     cave_mat(x1(i+1),x2(i))=1;
37 end
38 cave_mat(x1(1),x2(m))=1;
39 cave_mat(x2(m),x1(1))=1;
40 end

```

step1_randomgraph.m

```

1
2
3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4 %%% code adopted from %%
5 %%% Modeling and Simulating Social Systems with MATLAB %%
6 %%% http://www.soms.ethz.ch/teaching/MatlabFall2012 %%
7 %%% Authors: Stefan Brugger and Cristoph Schwirzer, 2011 %%
8 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9
10 function final= step1_randomgraph(n, p)
11 % Generates an undirected random graph (without self-loops) of size n (as
12 % described in the Erdoes-Renyi model)
13 %
14 % INPUT
15 % n: [1]: number of nodes
16 % p: [1]: probability that node i and node j, i != j, are connected by ...
    an edge
17 %
18 % OUTPUT
19 % final: [n n] full symmetric adjacency matrix representing the ...
    generated graph
20
21 % Note: A generation based on sprandsym(n, p) failed (for some values of p
22 % sprandsym was far off from the expected number of n*n*p non-zeros), ...
    therefore
23 % this longish implementation instead of just doing the following:
24 %
25 % B = sprandsym(n, p);
26 % A = (B-diag(diag(B))≠0);
27 %
28
29 % Idea: first generate the number of non-zero values in every row for a ...
    general

```

```

30 % 0-1-adjacency matrix. For every row this number is distributed ...
    binomially with
31 % parameters n and p.
32 %
33 % The following lines calculate "rowSize = binoinv(rand(1, n), n, p)", ...
    just in a
34 % faster way for large values of n.
35
36 % generate a vector of n values chosen u.a.r. from (0,1)
37 v = rand(1, n);
38 % Sort them and calculate the binomial cumulative distribution function with
39 % parameters n and p at values 0 to n. Afterwards match the sorted random
40 % 0-1-values to those cdf-values, i.e. associate a binomial distributed ...
    value
41 % with each value in r. Each value in v also corresponds to a value in r:
42 % permute the values in rowSize s.t. they correspond to the order given ...
    in v.
43 [r index] = sort(v); % i.e. v(index) == r holds
44 rowSize = zeros(1, n);
45 j = 0;
46 binoCDF = cumsum(binopdf(0:n, n, p));
47 for i = 1:n
48     while j < n && binoCDF(j+1) < r(i)
49         j = j + 1;
50     end
51     rowSize(i) = j;
52 end
53 rowSize(index) = rowSize;
54
55 % for every row choose the non-zero entries in it
56 nNZ = sum(rowSize);
57 I = zeros(1, nNZ);
58 J = zeros(1, nNZ);
59 j = 1;
60 for i = 1:n
61     I(j:j+rowSize(i)-1) = i;
62     J(j:j+rowSize(i)-1) = randsample(n, rowSize(i));
63     j = j + rowSize(i);
64 end
65
66 % restrict I and J to indices that correspond to entries above the main ...
    diagonal
67 % and finally construct a symmetric sparse matrix using I and J
68 upperTriu = find(I < J);
69 I = I(upperTriu);
70 J = J(upperTriu);
71 A = sparse([I;J], [J;I], ones(1, 2*size(I, 2)), n, n);
72 final = full(A);
73 end % random_graph(...)

```

step1_scalefree.m

```
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %%% code adopted from %%%
3 %%% Modeling and Simulating Social Systems with MATLAB %%%
4 %%% http://www.soms.ethz.ch/teaching/MatlabFall2012 %%%
5 %%% Authors: Stefan Brugger and Cristoph Schwirzer, 2011 %%%
6 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7
8 function final = step1_scalefree(n, m0, m)
9 % Use the Barabasi-Albert model to generate a scale free graph of size n (as
10 % described in Albert-Laszlo Barabasi & Reka Albert: "Emergence of scaling
11 % in random networks")
12 %
13 % INPUT
14 % n: [1]: number of nodes
15 % m0: [1]: number of initially placed nodes
16 % m: [1]: number of nodes a new added node is connected to,  $1 \leq m < m_0$ 
17 %
18 % OUTPUT
19 % final: [n n] full symmetric adjacency matrix representing the ...
    generated graph
20
21 % Start with a graph of size m0 and add edges to this graph. Each of ...
    these m0
22 % nodes is connected to at least m nodes.
23 B = zeros(m0, m0);
24 for i = 1:m0
25     neighbors = randsample(m0-1, m);
26     neighbors = neighbors + (neighbors >= i);
27     B(i,neighbors) = 1;
28     B(neighbors,i) = 1;
29 end
30
31 % Create a vector of edges added so far, i.e. nodes edge(2*i) and ...
    edge(2*i-1),
32 %  $1 \leq i \leq nEdges$ , are connected by an edge.
33 [rows, columns] = find(triu(B));
34 nEdges = size(rows, 1);
35 edges = reshape([rows';columns'], 2*nEdges, 1);
36 edges = [edges; zeros(2*(n-m0)*m,1)];
37
38 % Add nodes m0+1:n, one at a time. Each node is connected to m existing ...
    nodes,
39 % where each of the existing nodes is chosen with a probability that is
40 % proportional to the number of nodes it is already connected to.
41 used = zeros(n, 1); % is a node already used in a timestep?
42 for i = m0+1:n
43     neighbors = zeros(1, m);
```

```

44     for j=1:m
45         k = edges(randi(2*nEdges));
46         while used(k)
47             k = edges(randi(2*nEdges));
48         end
49         used(k) = 1;
50         neighbors(j) = k;
51     end
52     used(neighbors) = 0;
53     edges(2*nEdges+1:2*nEdges+2*m) = reshape([repmat(i, 1, m); ...
54         neighbors], ...
55         1, 2*m);
56     nEdges = nEdges+m;
57 end
58 % finally construct a symmetric adjacency matrix using the vector of edges
59 edges = edges(1:2*nEdges);
60 first = edges(1:2:end);
61 second = edges(2:2:end);
62 A = sparse([first;second], [second;first], ones(2*nEdges, 1), n, n);
63 final=full(A);
64 end % scale_free(...)

```

step1_smallworld.m

```

1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %%% code adopted from %%%
3  %%% Modeling and Simulating Social Systems with MATLAB %%%
4  %%% http://www.soms.ethz.ch/teaching/MatlabFall2012 %%%
5  %%% Authors: Stefan Brugger and Cristoph Schwirzer, 2011 %%%
6  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7
8  function final = step1_smallworld(n, k, beta)
9  % Generate a small world graph using the "Watts and Strogatz model" as
10 % described in Watts, D.J.; Strogatz, S.H.: "Collective dynamics of
11 % 'small-world' networks."
12 % A graph with  $n*k/2$  edges is constructed, i.e. the nodal degree is  $n*k$  for
13 % every node.
14 %
15 % INPUT
16 % n: [1]: number of nodes of the graph to be generated
17 % k: [1]: mean degree (assumed to be an even integer)
18 % beta: [1]: rewiring probability
19 %
20 % OUTPUT
21 % A: [n n] sparse symmetric adjacency matrix representing the generated ...
    graph
22
23 % Construct a regular lattice: a graph with n nodes, each connected to k

```

```

24 % neighbors, k/2 on each side.
25 kHalf = k/2;
26 rows = reshape(repmat((1:n)', 1, k), n*k, 1);
27 columns = rows+reshape(repmat([(1:kHalf) (n-kHalf:n-1)], n, 1), n*k, 1);
28 columns = mod(columns-1, n) + 1;
29 B = sparse(rows, columns, ones(n*k, 1));
30 A = sparse([], [], [], n, n);
31
32 % With probability beta rewire an edge avoiding loops and link duplication.
33 % Until step i, only the columns 1:i are generated making implicit use ...
    of A's
34 % symmetry.
35 for i = 1:n
36     % The i-th column is stored full for fast access inside the ...
        following loop.
37     col= [full(A(i, 1:i-1))'; full(B(i:end, i))];
38     for j = i+find(col(i+1:end))'
39         if (rand()<beta)
40             col(j)=0;
41             k = randi(n);
42             while k==i || col(k)==1
43                 k = randi(n);
44             end
45             col(k) = 1;
46         end
47     end
48     A(:,i) = col;
49 end
50
51 % A is not yet symmetric: to speed things up, an edge connecting i and ...
    j, i < j
52 % implies A(i,j)==1, A(i,j) might be zero.
53 T = triu(A);
54 A = T+T';
55 final=sparse(A);
56
57 end % small_world(...)

```

A.3 Step 2: Rewiring process

step2.m

```

1 function [mat, vec, dominant_freq, most_freq] = ...
    step2(t_end,phi,alpha,mat,vec,p,threshold)
2 %%%%%%%%%%%%%%% step2: Runing the Simulation %%%%%%%%%%%%%%%
3 %%% t_end: number of iterations
4 %%% phi: network reorganization rate
5 %%% alpha: innovation rate

```

```

6  %%% mat: initial connectivity matrix
7  %%% vec: initial idea vector
8  %%% p: initial number of opinions
9  %%% then it outputs:
10 %%% mat: connectivity matrix after simulation
11 %%% vec: idea vector after simulation
12 %%% dominant_freq: the vector holding the frequency of dominant idea
13 %%% most_freq: the vector holding the index of dominating idea in each time
14
15 most_freq=zeros(1,t_end); %%% Vector for storing the index of the ...
    dominating idea in each time step.
16 dominant_freq=zeros(1,t_end); %%% Vector for storing the frequency of ...
    dominant idea in each time step.
17 a=size(mat);
18 n=a(1); %%% number of agents
19
20 for t=1:t_end
21     x1=ceil(rand()*n); %%% choosing one person randomly for network ...
        reorganization or changing idea
22     a1=rand();
23     b1=phi;
24     if a1<b1 %%% i.e with probability phi to reorganize the network
25         v00=find(mat(x1,:)==1); %%% defining neighbours of x1
26         v=find(vec(v00)~=vec(x1)); %%% define neighbours of x1 that do not ...
            have the same idea as x1
27         if ~isempty(v)
28             x2=v(ceil(rand()*length(v))); %%% choosing one neighbour with ...
                different idea randomly to remove connection with
29             mat(x1,x2)=0; %%% deletion of the edge between x1 and x2
30             mat(x2,x1)=0; %%% deletion of the edge between x2 and x1
31             similar_idea=find(vec==vec(x1)); %%% define the agents with ...
                the same idea as x1
32             non_neighbor_sim_idea=setdiff(similar_idea,v00); %%% the ...
                agents with similar idea as x1 who are not neighbor of x1
33             if ~isempty(non_neighbor_sim_idea)
34                 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
35                 mat(x1,x3)=1; %%% formation of new edge between x1 and x3
36                 mat(x3,x1)=1; %%% formation of new edge between x3 and x1
37             else %%% we'll forget about network reorganization in this ...
                time by reforming the deleted edges.
38                 mat(x1,x2)=1;
39                 mat(x2,x1)=1;
40             end
41         end
42
43     else %%% otherwise change the idea of x1 to that of one of it's ...
        randomly chosen neighbours

```



```

44     v2=find(mat(x1,')==1); %% defining the neighbours of x1
45     if ~isempty(v2)
46         vv=vec(v2); %% the corresponding ideas of the neighbours
47         vvv=unique(vv); %% vector of all the distinct ideas
48         freq=zeros(1,length(vvv)); %% vector for frequencies of the ideas
49         for i=1:length(vvv) %% to test for all distinct ideas
50             if length(find(vv==vvv(i)))>(threshold*length(v2)) ...
51                 %%whether the frequency is larger than the threshold ...
52                 [To include complex contagion definition]
53                 freq(i)=1;
54             end
55         end
56         candidates=vvv(find(freq==1)); %% The ideas meeting the threshold
57         candidates_size=length(candidates); %% The number of ideas ...
58         meeting the threshold
59         if candidates_size>0
60             chosen=ceil(rand()*candidates_size); %% Randomly choose one ...
61             of the candidates
62             vec(x1)=candidates(chosen); %% change the idea of x1 to ...
63             the chosen idea
64         end
65     end
66 end
67
68 y=ceil(rand()*n); %%choosing one person randomly for coming up with ...
69 a new idea
70 a2=rand();
71 b2=alpha;
72 if a2<b2 %% i.e. with probability alpha to generate a novel idea
73     bound=10^6; %% to limit the index of new ideas to 10^6 which ...
74     simulates nearly boundryless pool of ideas
75     new_idea=ceil(rand()*bound)+p; %% the index of new idea
76     vec(y)=new_idea; %% changing the idea of agent y to the novel one
77 end
78
79 %%%
80 most_freq(t)=mode(vec); %% the dominant idea at time t
81 dominant_freq(t)=length(find(vec==most_freq(t))); %% the frequency ...
82 of the dominant idea at time t
83 end
84 end

```

A.4 Step 3: Results

step3a.m

```

1 function n_index = step3a(mat,vec)
2 %%%%%%%%% step3a: defining the neighbourhood index between similar ideas ...
3 %%%%%%%%%

```

```

3  %% mat: adjacency matrix
4  %% vec: idea vector
5  %% it outputs: n_index which is the average neighbour index of the network
6  u=unique(vec); %% collection of distinct ideas
7  w=zeros(1,length(u)); %% initializing the vector for storing the ...
    neighbourhood index for each distinct idea
8  s=zeros(1,length(u)); %% initializing the vector for storing the number ...
    of agents holding each distinct ideas
9  for i=1:length(u)
10     x=find(vec==u(i)); %% defining the set of agents holding the idea u(i)
11     s(i)=length(x);
12     if s(i)>1
13         sum1=0; %% number of agents with idea i which are in direct ...
            neighbourhood
14         for k=1:(s(i)-1) %% These two for loops are used to test the ...
            neighborhoods of all distinct pairs
15             for j=(k+1):s(i)
16                 if mat(x(k),x(j))==1 %% to check if they are neighbours
17                     sum1=sum1+1;
18                 end
19             end
20         end
21         max_neighbors=(s(i)*(s(i)-1)/2); %% normalizing factor (i.e., the ...
            maximum number of pairs of distinct agents)
22         w(i)=sum1/max_neighbors; %% neighborhood index for idea 'i'
23     else
24         w(i)=0;
25     end
26 end
27 norm1=sum(s); %% Normalizing factor
28 count=sum(w.*s); %% The weighted sum of neighborhood index
29 n_index=count/norm1; %% average neighbour index of the whole network
30 end

```

steb3b.m

```

1  function intra_idea_distance = step3b(mat,vec,s,c)
2  %%%%%%%%%%% step3b: Defining the average of the average shortest ...
    distance between agents holding the same idea %%%%%%%%%%%
3  %% mat: adjacency matrix
4  %% vec: idea vector
5  %% s: number of connected components
6  %% c: vector which assigns each node to a connected component
7  %% and it outputs the average of average intra_distance between agents
8  %% holding the same idea %%%%%%%%%%%
9  unq=unique(vec); %% collection of distinct ideas
10 mean_sum=zeros(1,length(unq)); %% initializing the vector for storing ...
    the average distance between agents holding a particular idea

```

```

11 n=zeros(1,length(unq)); %%% initializing the corresponding vector for ...
    storing the number of agents holding a certain idea
12 mats=sparse(mat);
13
14 for i=1:length(unq)
15     x=find(vec==unq(i)); %%% defining the set of agents holding the ...
        idea unq(i)
16     n(i)=length(x); %%% number of agents holding the idea unq(i)
17     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
18     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
19     for m=1:s %%% for each connected component
20         y=x(find(c(x)==m)); %%% the set of agents holding the idea ...
            unq(i) who belong to the m-th connected component
21         nn(m)=length(y); %%% number of agents holding the idea unq(i) ...
            who belong to the m-th connected component
22         sum0=0;
23         if nn(m)>1
24             for j=1:(nn(m)-1)
25                 for k=(j+1):nn(m)
26                     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]
27                     sum0=sum0+dis; %%% sum of the distances between ...
                        agents holding the idea unq(i) who belong to m-th ...
                        connected component
28                 end
29             end
30             norml0=nn(m)*(nn(m)-1)/2; %%% normalizing factor
31             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
32         else
33             mean_sum0(m)=0;
34         end
35     end
36
37     if n(i)>1
38         mean_sum(i)=(sum(mean_sum0.*nn))/(sum(nn)); %%% The avarage ...
            intra-idea distance between agents holding the idea unq(i)
39     else
40         mean_sum(i)=0;
41     end
42 end
43

```

```

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

```

step3e.m

```

1 function [ average_dominance_time ] = step3e( most_freq )
2 %%%% this function calculates the average of dominance time for the ...
    dominating idea during the simulation
3 %%%% most_freq: is the vector obtained from simulation which holds the ...
    index of dominating ideas for each time steps of the simulation
4 %%%% it outputs the average_time which is the average of the dominance ...
    time for different dominance periods
5 temp=zeros(1,length(most_freq)); %% a temporary array which will hold ...
    the dominating period for dominating idea
6 count=1; %% is the number of consecutive time steps in which an idea is ...
    considered as dominating
7 ind=0; %% is the index of domination period
8 for i=2:length(most_freq)
9     if most_freq(i)==most_freq(i-1)
10         count=count+1; %% the number of consecutive steps is conted
11     else
12         ind=ind+1; %% as soon as another domination period gets tarted ...
            the index of domination period adds by one
13         temp(ind)=count; %% the number of consecutive steps will be ...
            stored in the ind-th index of tmp
14         count=0; %% and count will be reset to zero to count the ...
            duration of the new domination period.
15     end
16 end
17
18 if ind==0 %% in case during the simulation, just one special idea ...
    remains dominating forever
19     ind=ind+1;
20     temp(ind)=count;
21 end
22
23 %% 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
24 final=zeros(1,ind);
25 for j=1:ind
26     final(j)=temp(j);
27 end
28 average_dominance_time=mean(final);
29 end

```

step4a.m

```

1
2 function clust_coeff = step4a( mat )
3 % This function calculates the clustering coefficient of a network with
4 % Corresponding connectivity matrix: mat
5 % The approach used for calculation of clustering coefficient was the one ...
   which was proposed by Watts and Strogatz
6 % in which the clustering coefficient of the whole network equals to the
7 % average of the local clustering coefficient of all nodes:
8 % D. J. Watts and Steven Strogatz (June 1998). "Collective dynamics of ...
   'small-world' networks". Nature 393 (6684): 440 442 .
9 a=size(mat);
10 local_clust_coeff=zeros(1,a(1)); %%% the vector to store the local ...
   clustering coefficient of each node of the network
11 for i=1:a(1) %%% for each agent in the network
12     x=find(mat(i,:)==1); %%% find all the neighbours of agent i
13     count=0;
14     l=length(x);
15     if l>1
16         for j=1:(l-1) %%% loop to investigate the neighbour relationship for ...
           all possible pairs among the neighbors of node i
17             for k=(j+1):l
18                 if mat(x(j),x(k))==1
19                     count=count+1; % to count the number of neighbour ...
                       relationships
20             end
21         end
22     end
23     norml=l*(l-1)/2; % normalizing factor (i.e. the number of distinct ...
   pairs)
24     local_clust_coeff(i)=count/norml; % local clustering coefficient of ...
   node i
25     else
26         local_clust_coeff(i)=0;
27     end
28 end
29 clust_coeff=sum(local_clust_coeff)/a(1); % the average clustering ...
   coefficient of entire network
30 end

```

step4b.m

```

1 function [dgr,frq] = step4b(mat)
2 %%% this function outputs the degree vector (dgr) and its corresponding
3 %%% frequency vector (frq) from the connectivity matrix (mat). In other ...
   words,
4 %%% first the degree of all nodes in the network will be calculated and
5 %%% then the set of unique degrees will be stored in the dgr vector and the
6 %%% corresponding frequency will be stored in vector frq. For example,

```

```

7  %% degree x will be stored in the i-th element of the dgr vector, then ...
   the number of nodes whose degree equals x
8  %% will be calculated and will be stored in the i-th element of vector frq.
9  a=size(mat);
10 degr=zeros(1,a(1)); %% the vector which stores the degree of each node ...
    of the network
11 for i=1:a(1)
12     degr(i)=sum(mat(i,:)); %% degree for each node
13 end
14 dgr=unique(degr); %% set of unique degrees of the nodes of the network
15 frq=zeros(1,length(dgr)); %% the corresponding frequency vector
16 for j=1:length(dgr)
17     frq(j)=length(find(degr==dgr(j))); %% frq(j) is the number of nodes ...
        in the network whose degree equals to dgr(j)
18 end
19
20 end

```

step4d.m

```

1  function average_path_length = step4d( mat,s,c )
2  %This function outputs the average path length for the graph
3  %mat: connectivity matrix
4  %s: number of connected components
5  %c: the connected component vector
6  av_p_l=zeros(1,s); %% the vector which will store the average path ...
    length for each connected component of the 'mat' network
7  n=zeros(1,s); %% the vector which will store the number of nodes ...
    belonging to each connected component.
8  sp=sparse(mat);
9
10 for i=1:s %% for each connected component
11     x=find(c==i); %% characterizing the nodes which belong to the i-th ...
        connected component
12     a=length(x); %% the number of nodes in the i-th connected component
13     n(i)=a;
14     count=0;
15     if a>1
16         for k=1:(a-1)
17             for j=(k+1):a
18                 count=count+graphshortestpath(sp,x(k),x(j)); %% sum of the path ...
                    length between all pairs of node
19             end
20         end
21         norml=a*(a-1)/2; %% normalization factor
22         av_p_l(i)=count/norml; %% average of path length
23     else
24         av_p_l(i)=0;
25     end

```

```

26 end
27 average_path_length=sum(av_p_l.*n)/sum(n); %% weighted average path length
28 end

```

step4e.m

```

1 function graph.diameter = step4e( mat,s,c )
2 %This function outputs the diameter of the graph 'mat'
3 %mat: connectivity matrix
4 %s: number of connected components
5 %c: the connected component vector
6 sp=sparse(mat);
7 max_path=zeros(1,s); %% vector storing diameter for each connected ...
   components.
8 for i=1:s %%% for each connected component
9     x=find(c==i); %%% characterizing the nodes which belong to the i-th ...
   connected component
10    a=length(x); %%% the number of nodes in the i-th connected component
11    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
12    count=0;
13    if a>1
14        for k=1:(a-1)
15            for j=(k+1):a
16                count=count+1;
17                path_length(count)=graphshortestpath(sp,x(k),x(j)); %%% the ...
   vector storing the shortest path between pairs k and j
18            end
19        end
20        max_path(i)=max(path_length); % the maximum shortest path for ...
   connected component i
21    else
22        max_path(i)=0;
23    end
24 end
25 graph.diameter=max(max_path); % the maximum shortest path for the hole ...
   network (Diameter of the network)
26 end

```

A.5 Plotes code

plots_phase1.m

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Plots of phase 1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %%
3 %% Extracting information from the saved Mat files %%%%%%%%%

```

```

4 neighbor_ind=zeros(1,108);
5 int_id_dis=zeros(1,108);
6 nov_ind=zeros(1,108);
7 av_dom_tim=zeros(1,108);
8 dom_freq=zeros(108,1000);
9 alpha=[0.01,0.05,0.1];
10 phi=[0.1,0.3,0.5];
11 threshold=[0.001,0.01,0.05];
12 count=0;
13 f=figure();
14 for i=1:4
15     switch i
16         case 1
17             s1='Caveman';
18         case 2
19             s1='Random';
20         case 3
21             s1='Scale_free';
22         case 4
23             s1='Small_world';
24     end
25     for j=1:3
26         s2=int2str(j);
27         for k=1:3
28             s3=int2str(k);
29             for l=1:3
30                 count=count+1;
31                 s4=int2str(l);
32                 name=['phase1_',s1,'_',s2,'_',s3,'_',s4];
33                 a=load(name);
34                 neighbor_ind(count)=a.neighbor_index;
35                 int_id_dis(count)=a.intra_idea_distance;
36                 nov_ind(count)=a.nov_index;
37                 av_dom_tim(count)=a.average_dominance_time;
38                 dom_freq(count,:)=a.dominant_freq;
39             end
40         end
41     end
42 end
43
44 %% phase diagrams for each parameter pairs for the results of the effects
45 %% of network structure on idea distribution.
46
47 for i=1:4
48     switch i
49         case 1
50             s1='Caveman';
51             val=0;
52         case 2
53             s1='Random';

```



```

54         val=27;
55         case 3
56             s1='Scale free';
57             val=54;
58         case 4
59             s1='Small world';
60             val=81;
61     end
62     for j=1:4
63         switch j
64             case 1
65                 s2='neighbor index';
66                 main_vec=neighbor_ind;
67             case 2
68                 s2='intra idea distance';
69                 main_vec=int_id_dis;
70             case 3
71                 s2='nov index';
72                 main_vec=nov_ind;
73             case 4
74                 s2='average dominance time';
75                 main_vec=av_dom_tim;
76         end
77         for k=1:3
78             switch k
79                 case 1
80                     s3='(alpha:phi)';
81                     x=0:0.5:1;
82                     xl='alpha';
83                     y=0:0.5:1;
84                     yl='phi';
85                     for l=1:3
86                         M=[main_vec(val+1),main_vec(val+1+9),main_vec(val+1+18);main_vec(val+1+3),ma
87                         %
88                         [xlab,ylab]=meshgrid(x,y);
89                         hold on;
90                         view(0,90);
91                         surf(xlab,ylab,M,'EdgeColor','none');
92                         colorbar;
93                         set(gca,'FontSize',14)
94                         xlabel(xl);
95                         ylabel(yl);
96                         switch l
97                             case 1
98                                 s5='first';
99                             case 2
100                                 s5='second';
101                             case 3
102                                 s5='third';
103                     end

```

```

104         name2=['phase diagram of ',s2,' for alpha versus phi',' ...
               and','the', s5,' threshold','obtained ...
               for',s1,'network structure'];
105     title(name2);
106     name=['plotting1-', 'Pdiag-',s1,'-',s2,'-',s3,'-',s4,'AND','the', ...
           s5,' threshold'];
107     saveas(f,name);
108     hold off
109     %
110 end
111 case 2
112     s3='(alpha:threshold)';
113     x=0:0.5:1;
114     xl='alpha';
115     y=0:0.5:1;
116     yl='threshold';
117     for l=1:3
118         M=[main_vec(val+9*(l-1)+1),main_vec(val+9*(l-1)+2),main_vec(val+9*(l-1)+3);ma
119           %
120           [xlab,ylab]=meshgrid(x,y);
121           hold on;
122           view(0,90);
123           surf(xlab,ylab,M,'EdgeColor','none');
124           colorbar;
125           set(gca,'FontSize',14)
126           xlabel(xl);
127           ylabel(yl);
128           switch l
129               case 1
130                   s5='first';
131               case 2
132                   s5='second';
133               case 3
134                   s5='third';
135           end
136           name2=['phase diagram of ',s2,' for alpha versus ...
                 threshold',' and','the', s5,' phi','obtained ...
                 for',s1,'network structure'];
137       title(name2);
138       name=['plotting1-', 'Pdiag-',s1,'-',s2,'-',s3,'-',s4,' ...
             and','the', s5,' phi'];
139       saveas(f,name);
140       hold off
141       %
142 end
143
144 case 3
145     s3='(phi:threshold)';
146     x=0:0.5:1;
147     xl='phi';

```

```

148         y=0:0.5:1;
149         yl='threshold';
150         for l=1:3
151             M=[main_vec(val+3*(l-1)+1),main_vec(val+3*(l-1)+2),main_vec(val+3*(l-1)+3);ma
152             %
153             [xlab,ylab]=meshgrid(x,y);
154             hold on;
155             view(0,90);
156             surf(xlab,ylab,M,'EdgeColor','none');
157             colorbar;
158             set(gca,'FontSize',14)
159             xlabel(xl);
160             ylabel(yl);
161             switch l
162                 case 1
163                     s5='first';
164                 case 2
165                     s5='second';
166                 case 3
167                     s5='third';
168             end
169             name2=['phase diagram of ',s2,' for phi versus ...
170                 threshold',' and ','the', s5,' threshold','obtained ...
171                 for',s1,'network structure'];
172             title(name2);
173             name=['plotting1_', 'Pdiag_',s1,'_',s2,'_',s3,'_',s4,' ...
174                 and','the', s5,' threshold'];
175             saveas(f,name);
176             hold off
177             %
178         end
179     end
180 end
181 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
182 %%% Dependence of the property of each network to the parameters
183
184 for i=1:4
185     switch i
186         case 1
187             s1='Caveman';
188             val=0;
189         case 2
190             s1='Random';
191             val=27;
192         case 3
193             s1='Scale free';
194             val=54;

```

```

195         case 4
196             s1='Small world';
197             val=81;
198     end
199     for j=1:4
200         switch j
201             case 1
202                 s2='neighborhood index';
203                 main_vec=neighbor_ind;
204             case 2
205                 s2='intra idea distance';
206                 main_vec=int_id_dis;
207             case 3
208                 s2='novelty index';
209                 main_vec=nov_ind;
210             case 4
211                 s2='average dominance time';
212                 main_vec=av_dom_tim;
213         end
214         for k=1:3
215             switch k
216                 case 1
217                     s3='threshold';
218                     f=figure();
219                     for l=1:3
220                         line=main_vec(3*(0:8)+val+1);
221                         set(gca,'FontSize',14)
222                         xlabel('Different pairs of alpha and phi values');
223                         ylabel(s2);
224                         namek=['Dependence of',' ',s2,' ','on',' ',s3,' ...
225                             ','for',' ',s1,' ','network structure'];
226                         title(namek);
227                     switch l
228                         case 1
229                         plot(line,'—ks','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','k');
230                         hold on
231                         case 2
232                         plot(line,'—bs','LineWidth',2,'MarkerEdgeColor','b','MarkerFaceColor','b');
233                         hold on
234                         case 3
235                         plot(line,'—rs','LineWidth',2,'MarkerEdgeColor','r','MarkerFaceColor','r');
236                     end
237                     legend('threshold=0.001','threshold=0.01','threshold=0.05')
238                     name=['plotting1-',s1,'-',s2,'-', 'threshold_sensitivity'];
239                     saveas(f,name);
240                 end
241             end
242         case 2
243             s3='alpha';

```

```

244         f=figure();
245         for l=1:3
246             line=main_vec([1,2,3,10,11,12,19,20,21]+val+3*(l-1));
247             set(gca,'FontSize',14)
248             xlabel('Different pairs of threshold and phi ...
                values');
249             ylabel(s2);
250             namek=['Dependence of',' ',s2,' ','on',' ',s3,' ...
                ','for',' ',s1,' ','network structure'];
251             title(namek);
252
253             switch l
254                 case 1
255                     plot(line,'—ks','LineWidth',2,'MarkerEdgeColor','k','MarkerF
256                     hold on
257                 case 2
258                     plot(line,'—bs','LineWidth',2,'MarkerEdgeColor','b','MarkerF
259                     hold on
260                 case 3
261                     plot(line,'—rs','LineWidth',2,'MarkerEdgeColor','r','MarkerF
262             end
263             legend('alpha=0.01','alpha=0.05','alpha=0.10')
264             name=['plotting1-',s1,'-',s2,'-', 'alpha.sensitivity'];
265             saveas(f,name);
266         end
267
268     case 3
269         s3='phi';
270         f=figure();
271         for l=1:3
272             line=main_vec((1:9)+val+9*(l-1));
273             set(gca,'FontSize',14)
274             xlabel('Different pairs of threshold and alpha ...
                values');
275             ylabel(s2);
276             namek=['Dependence of',' ',s2,' ','on',' ',s3,' ...
                ','for',' ',s1,' ','network structure'];
277             title(namek);
278
279             switch l
280                 case 1
281                     plot(line,'—ks','LineWidth',2,'MarkerEdgeColor','k','MarkerF
282                     hold on
283                 case 2
284                     plot(line,'—bs','LineWidth',2,'MarkerEdgeColor','b','MarkerF
285                     hold on
286                 case 3
287                     plot(line,'—rs','LineWidth',2,'MarkerEdgeColor','r','MarkerF
288             end
289             legend('phi=0.1','phi=0.3','phi=0.5')

```

```

290             name=['plotting1-',s1,'-',s2,'-', 'phi_sensitivity'];
291             saveas(f,name);
292         end
293     end
294 end
295 end
296 end
297
298 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Plotting each network feature for each set of
299 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% parameters %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
300 for i=1:4
301     switch i
302     case 1
303         f=figure();
304         line1=neighbor_ind(1:27);
305         line2=neighbor_ind(28:54);
306         line3=neighbor_ind(55:81);
307         line4=neighbor_ind(82:108);
308         plot(line1,'--ks','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','k','MarkerSize',10);
309         hold on
310         plot(line2,'--gs','LineWidth',2,'MarkerEdgeColor','g','MarkerFaceColor','g','MarkerSize',10);
311         hold on
312         plot(line3,'--bs','LineWidth',2,'MarkerEdgeColor','b','MarkerFaceColor','b','MarkerSize',10);
313         hold on
314         plot(line4,'--rs','LineWidth',2,'MarkerEdgeColor','r','MarkerFaceColor','r','MarkerSize',10);
315         set(gca,'FontSize',14)
316         xlabel('Different triplets of alpha ,phi and threshold values')
317         ylabel('Neighborhood index')
318         title('The influence of network structure on neighborhood ...
319             index');
320         legend('Caveman Network','Random Network','Scale Free ...
321             Network','Small World Network')
322         name=('Plotting1_Network_Comparison_Neighborhood_index');
323         saveas(f,name);
324     case 2
325         f=figure();
326         line1=int_id_dis(1:27);
327         line2=int_id_dis(28:54);
328         line3=int_id_dis(55:81);
329         line4=int_id_dis(82:108);
330         plot(line1,'--ks','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','k','MarkerSize',10);
331         hold on
332         plot(line2,'--gs','LineWidth',2,'MarkerEdgeColor','g','MarkerFaceColor','g','MarkerSize',10);
333         hold on
334         plot(line3,'--bs','LineWidth',2,'MarkerEdgeColor','b','MarkerFaceColor','b','MarkerSize',10);
335         hold on
336         plot(line4,'--rs','LineWidth',2,'MarkerEdgeColor','r','MarkerFaceColor','r','MarkerSize',10);
337         set(gca,'FontSize',14)
338         xlabel('Different triplets of alpha ,phi and threshold values')
339         ylabel('Intra-idea distance')

```

```

338         title('The influence of network structure on intra-idea ...
           distance')
339         legend('Caveman Network','Random Network','Scale Free ...
           Network','Small World Network')
340         name=('Plotting1_Network Comparison_intra-idea-distance');
341         saveas(f,name);
342     case 3
343         f=figure();
344         line1=nov_ind(1:27);
345         line2=nov_ind(28:54);
346         line3=nov_ind(55:81);
347         line4=nov_ind(82:108);
348         plot(line1,'—ks','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','k','MarkerSize',10)
349         hold on
350         plot(line2,'—gs','LineWidth',2,'MarkerEdgeColor','g','MarkerFaceColor','g','MarkerSize',10)
351         hold on
352         plot(line3,'—bs','LineWidth',2,'MarkerEdgeColor','b','MarkerFaceColor','b','MarkerSize',10)
353         hold on
354         plot(line4,'—rs','LineWidth',2,'MarkerEdgeColor','r','MarkerFaceColor','r','MarkerSize',10)
355         set(gca,'FontSize',14)
356         xlabel('Different triplets of alpha ,phi and threshold values')
357         ylabel('Novelty Index')
358         title('The influence of network structure on Novelty index')
359         legend('Caveman Network','Random Network','Scale Free ...
           Network','Small World Network')
360         name=('Plotting1_Network Comparison_intra-novelty_index');
361         saveas(f,name);
362     case 4
363         f=figure();
364         line1=av_dom_tim(1:27);
365         line2=av_dom_tim(28:54);
366         line3=av_dom_tim(55:81);
367         line4=av_dom_tim(82:108);
368         plot(line1,'—ks','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','k','MarkerSize',10)
369         hold on
370         plot(line2,'—gs','LineWidth',2,'MarkerEdgeColor','g','MarkerFaceColor','g','MarkerSize',10)
371         hold on
372         plot(line3,'—bs','LineWidth',2,'MarkerEdgeColor','b','MarkerFaceColor','b','MarkerSize',10)
373         hold on
374         plot(line4,'—rs','LineWidth',2,'MarkerEdgeColor','r','MarkerFaceColor','r','MarkerSize',10)
375         set(gca,'FontSize',14)
376         xlabel('Different triplets of alpha ,phi and threshold values')
377         ylabel('Average Dominance Time')
378         title('The influence of network structure on Average ...
           Dominance Time')
379         legend('Caveman Network','Random Network','Scale Free ...
           Network','Small World Network')
380         name=('Plotting1_Network Comparison_Average Dominance Time');
381         saveas(f,name);
382     end

```

```

383 end
384
385
386 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% plotting frequency of dominant idea over time for each
387 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% parameter sets %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
388 count=0;
389 for choice1=1:3
390     switch choice1
391         case 1
392             s1='phi=0.1';
393             m1='1';
394         case 2
395             s1='phi=0.3';
396             m1='2';
397         case 3
398             s1='phi=0.5';
399             m1='3';
400     end
401
402     for choice2=1:3
403         switch choice2
404             case 1
405                 s2='alpha=0.01';
406                 m2='1';
407             case 2
408                 s2='alpha=0.05';
409                 m2='2';
410             case 3
411                 s2='alpha=0.1';
412                 m2='3';
413         end
414
415         for choice3=1:3
416             switch choice3
417                 case 1
418                     s3='threshold=0.001';
419                     m3='1';
420                 case 2
421                     s3='threshold=0.01';
422                     m3='2';
423                 case 3
424                     s3='threshold=0.05';
425                     m3='3';
426             end
427             count=count+1;
428             line1=dom_freq(count,:);
429             line2=dom_freq(27+count,:);
430             line3=dom_freq(54+count,:);
431             line4=dom_freq(81+count,:);
432             f=figure();

```



```

433     plot(line1,'--ks','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','k','Marke
434     hold on
435     plot(line2,'--gs','LineWidth',2,'MarkerEdgeColor','g','MarkerFaceColor','g','Marke
436     hold on
437     plot(line3,'--bs','LineWidth',2,'MarkerEdgeColor','b','MarkerFaceColor','b','Marke
438     hold on
439     plot(line4,'--rs','LineWidth',2,'MarkerEdgeColor','r','MarkerFaceColor','r','Marke
440     set(gca,'FontSize',14)
441     xlabel('time')
442     ylabel('Frequency of the dominant idea')
443     title(['The influence of network structure on the frequency ...
         of the dominant idea',' ',s1,' ',s2,' ',s3]);
444     legend('Caveman Network','Random Network','Scale Free ...
         Network','Small World Network')
445     name=(['Plotting1Network ...
         Comparison-frequency-dominant-idea-',m1,'_',m2,'_',m3]);
446     saveas(f,name);
447 end
448 end
449 end

```

plots_phase2.m

```

1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Plots of phase 2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %%
3  %% Extracting information from the saved Mat files %%%%%%%%%
4  %%
5  clust_coefficient=zeros(1,81);
6  s=zeros(1,81);
7  average_path_length=zeros(1,81);
8  diam=zeros(1,81);
9  dgr=cell(1,81);
10 frq=cell(1,81);
11 alpha=[0.01,0.05,0.1];
12 phi=[0.1,0.3,0.5];
13 threshold=[0.001,0.01,0.05];
14
15 count=0;
16
17 for i=1:3
18     switch i
19         case 1
20             s1='Random';
21         case 2
22             s1='Parallel';
23         case 3
24             s1='Antiparallel';
25     end
26     for j=1:3

```

```

27         s2=int2str(j);
28         for k=1:3
29             s3=int2str(k);
30             for l=1:3
31                 count=count+1;
32                 s4=int2str(l);
33                 name=['phase2_',s1,'_',s2,'_',s3,'_',s4];
34                 a=load(name);
35                 clust_coefficient(count)=a.clust_coefficient;
36                 s(count)=a.s;
37                 average_path_length(count)=a.average_path_length;
38                 diam(count)=a.diam;
39                 dgr{1,count}=a.dgr;
40                 frq{1,count}=a.frq;
41             end
42         end
43     end
44 end
45
46 f=figure();
47
48 %% phase diagrams for each parameter pairs for the results of the effects
49 %% of idea distribution on network structure.
50
51 for i=1:3
52     switch i
53         case 1
54             s1='Random';
55             val=0;
56         case 2
57             s1='Parallel';
58             val=27;
59         case 3
60             s1='Antiparallel';
61             val=54;
62     end
63     for j=1:4
64         switch j
65             case 1
66                 s2='Clusteing Coefficient';
67                 main_vec=clust_coefficient;
68             case 2
69                 s2='Number of connected components';
70                 main_vec=s;
71             case 3
72                 s2='Average path length';
73                 main_vec=average_path_length;
74             case 4
75                 s2='Network diameter';
76                 main_vec=diam;

```

```

77     end
78     for k=1:3
79         switch k
80             case 1
81                 s3='(alpha:phi)';
82                 x=0:0.5:1;
83                 xl='alpha';
84                 y=0:0.5:1;
85                 yl='phi';
86                 for l=1:3
87                     M=[main_vec(val+l),main_vec(val+l+9),main_vec(val+l+18);main_vec(val+l+3),ma
88                     %
89                     [xlab,ylab]=meshgrid(x,y);
90                     hold on;
91                     view(0,90);
92                     surf(xlab,ylab,M,'EdgeColor','none');
93                     colorbar;
94                     set(gca,'FontSize',14)
95                     xlabel(xl);
96                     ylabel(yl);
97                     switch l
98                         case 1
99                             s5='first';
100                        case 2
101                            s5='second';
102                        case 3
103                            s5='third';
104                    end
105                    name2=['phase diagram of ',s2,' for alpha versus phi',' ...
                        and','the', s5,' threshold','obtained for',s1,'idea ...
                        distribution'];
106                    title(name2);
107                    name=['Plotting2_', 'Pdiag-',s1,'-',s2,'-',s3,'-',s4,'AND','the', ...
                        s5,' threshold'];
108                    saveas(f,name);
109                    hold off
110                    %
111                end
112            case 2
113                s3='(alpha:threshold)';
114                x=0:0.5:1;
115                xl='alpha';
116                y=0:0.5:1;
117                yl='threshold';
118                for l=1:3
119                    M=[main_vec(val+9*(l-1)+1),main_vec(val+9*(l-1)+2),main_vec(val+9*(l-1)+3);ma
120                    %
121                    [xlab,ylab]=meshgrid(x,y);
122                    hold on;
123                    view(0,90);

```

```

124         surf(xlab,ylab,M,'EdgeColor','none');
125         colorbar;
126         set(gca,'FontSize',14)
127         xlabel(xl);
128         ylabel(yl);
129         switch l
130             case 1
131                 s5='first';
132             case 2
133                 s5='second';
134             case 3
135                 s5='third';
136         end
137         name2=['phase diagram of ',s2,' for alpha versus ...
                threshold',' and','the', s5,' phi','obtained ...
                for',s1,'idea distribution'];
138         title(name2);
139         name=['Plotting2_', 'Pdiag_',s1,'_',s2,'_',s3,'_',s4,' ...
                and','the', s5,' phi'];
140         saveas(f,name);
141         hold off
142         %
143     end
144
145     case 3
146         s3='(phi:threshold)';
147         x=0:0.5:1;
148         xl='phi';
149         y=0:0.5:1;
150         yl='threshold';
151         for l=1:3
152             M=[main_vec(val+3*(l-1)+1),main_vec(val+3*(l-1)+2),main_vec(val+3*(l-1)+3);ma
153             %
154             [xlab,ylab]=meshgrid(x,y);
155             hold on;
156             view(0,90);
157             surf(xlab,ylab,M,'EdgeColor','none');
158             colorbar;
159             set(gca,'FontSize',14)
160             xlabel(xl);
161             ylabel(yl);
162             switch l
163                 case 1
164                     s5='first';
165                 case 2
166                     s5='second';
167                 case 3
168                     s5='third';
169             end

```

```

170         name2=['phase diagram of ',s2,' for phi versus ...
                threshold',' and','the', s5,' threshold','obtained ...
                for',s1,'idea distribution'];
171         title(name2);
172         name=['Plotting2-','Pdiag-',s1,'-',s2,'-',s3,'-',s4,' ...
                and','the', s5,' threshold'];
173         saveas(f,name);
174         hold off
175         %
176     end
177 end
178 end
179 end
180 end
181
182 %%% Dependence of the property of each network to the parameters
183
184 for i=1:3
185
186     switch i
187     case 1
188         s1='Random';
189         val=0;
190     case 2
191         s1='Parallel';
192         val=27;
193     case 3
194         s1='Antiparallel';
195         val=54;
196     end
197
198     for j=1:4
199         switch j
200         case 1
201             s2='clust_coefficient';
202             main_vec=clust_coefficient;
203         case 2
204             s2='s';
205             main_vec=s;
206         case 3
207             s2='average_path_length';
208             main_vec=average_path_length;
209         case 4
210             s2='diam';
211             main_vec=diam;
212         end
213         for k=1:3
214             switch k
215             case 1
216                 s3='threshold';

```

```

217         f=figure();
218         for l=1:3
219             line=main_vec(3*(0:8)+val+1);
220             set(gca,'FontSize',14)
221             xlabel('Different pairs of alpha and phi values');
222             ylabel(s2);
223             namek=['Dependence of',' ',s2,' ','on',' ',s3,' ...
                ','for',' ',s1,' ','idea distribution'];
224             title(namek);
225
226             switch l
227                 case 1
228                     plot(line,'—ks','LineWidth',2,'MarkerEdgeColor','k','MarkerF
229                     hold on
230                 case 2
231                     plot(line,'—bs','LineWidth',2,'MarkerEdgeColor','b','MarkerF
232                     hold on
233                 case 3
234                     plot(line,'—rs','LineWidth',2,'MarkerEdgeColor','r','MarkerF
235             end
236             legend('threshold=0.001','threshold=0.01','threshold=0.05')
237             name=['plotting1-',s1,'-',s2,'-', 'threshold_sensitivity'];
238             saveas(f,name);
239         end
240
241     case 2
242         s3='alpha';
243         f=figure();
244         for l=1:3
245             line=main_vec([1,2,3,10,11,12,19,20,21]+val+3*(l-1));
246             set(gca,'FontSize',14)
247             xlabel('Different pairs of threshold and phi ...
                values');
248             ylabel(s2);
249             namek=['Dependence of',' ',s2,' ','on',' ',s3,' ...
                ','for',' ',s1,' ','idea distribution'];
250             title(namek);
251
252             switch l
253                 case 1
254                     plot(line,'—ks','LineWidth',2,'MarkerEdgeColor','k','MarkerF
255                     hold on
256                 case 2
257                     plot(line,'—bs','LineWidth',2,'MarkerEdgeColor','b','MarkerF
258                     hold on
259                 case 3
260                     plot(line,'—rs','LineWidth',2,'MarkerEdgeColor','r','MarkerF
261             end
262             legend('alpha=0.01','alpha=0.05','alpha=0.10')
263             name=['plotting1-',s1,'-',s2,'-', 'alpha_sensitivity'];

```

```

264         saveas(f,name);
265     end
266
267     case 3
268         s3='phi';
269         f=figure();
270         for l=1:3
271             line=main_vec((1:9)+val+9*(l-1));
272             set(gca,'FontSize',14)
273             xlabel('Different pairs of threshold and alpha ...
                values');
274             ylabel(s2);
275             namek=['Dependence of',' ',s2,' ','on',' ',s3,' ...
                ','for',' ',s1,' ','idea distribution'];
276             title(namek);
277
278             switch l
279                 case 1
280                     plot(line,'—ks','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','k');
281                     hold on
282                 case 2
283                     plot(line,'—bs','LineWidth',2,'MarkerEdgeColor','b','MarkerFaceColor','b');
284                     hold on
285                 case 3
286                     plot(line,'—rs','LineWidth',2,'MarkerEdgeColor','r','MarkerFaceColor','r');
287             end
288             legend('phi=0.1','phi=0.3','phi=0.5')
289             name=['plotting1-',s1,'-',s2,'-', 'phi.sensitivity'];
290             saveas(f,name);
291         end
292     end
293 end
294 end
295 end
296
297
298 %%%%%%%%% generating plots for each features of network structure for each
299 %%%%%%%%% parameter sets %%%%%%%%%
300
301 for i=1:4
302     switch i
303         case 1
304             f=figure();
305             line1=clust_coefficient(1:27);
306             line2=clust_coefficient(28:54);
307             line3=clust_coefficient(55:81);
308             plot(line1,'—ks','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','k');
309             hold on
310             plot(line2,'—gs','LineWidth',2,'MarkerEdgeColor','g','MarkerFaceColor','g');
311             hold on

```

```

312     plot(line3, '--bs', 'LineWidth', 2, 'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'b', 'MarkerSize', 10)
313     set(gca, 'FontSize', 14)
314     xlabel('Different triplets of alpha ,phi and threshold values')
315     ylabel('Clustering Coefficient')
316     title('The influence of idea distribution on clustering ...
        coefficient');
317     legend('Random distribution', 'Parallel ...
        distribution', 'Antiparallel distribution')
318     name=('Plotting2_idea.Comparison_clustcoeff');
319     saveas(f, name);
320 case 2
321     f=figure();
322     line1=s(1:27);
323     line2=s(28:54);
324     line3=s(55:81);
325     plot(line1, '--ks', 'LineWidth', 2, 'MarkerEdgeColor', 'k', 'MarkerFaceColor', 'k', 'MarkerSize', 10)
326     hold on
327     plot(line2, '--gs', 'LineWidth', 2, 'MarkerEdgeColor', 'g', 'MarkerFaceColor', 'g', 'MarkerSize', 10)
328     hold on
329     plot(line3, '--bs', 'LineWidth', 2, 'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'b', 'MarkerSize', 10)
330     set(gca, 'FontSize', 14)
331     xlabel('Different triplets of alpha ,phi and threshold values')
332     ylabel('Number of connected components')
333     title('The influence of idea distribution on the number of ...
        connected components')
334     legend('Random distribution', 'Parallel ...
        distribution', 'Antiparallel distribution')
335     name=('Plotting2_idea.Comparison_conncmp');
336     saveas(f, name);
337 case 3
338     f=figure();
339     line1=average_path_length(1:27);
340     line2=average_path_length(28:54);
341     line3=average_path_length(55:81);
342     plot(line1, '--ks', 'LineWidth', 2, 'MarkerEdgeColor', 'k', 'MarkerFaceColor', 'k', 'MarkerSize', 10)
343     hold on
344     plot(line2, '--gs', 'LineWidth', 2, 'MarkerEdgeColor', 'g', 'MarkerFaceColor', 'g', 'MarkerSize', 10)
345     hold on
346     plot(line3, '--bs', 'LineWidth', 2, 'MarkerEdgeColor', 'b', 'MarkerFaceColor', 'b', 'MarkerSize', 10)
347     set(gca, 'FontSize', 14)
348     xlabel('Different triplets of alpha ,phi and threshold values')
349     ylabel('average path length')
350     title('The influence of idea distribution on average path ...
        length')
351     legend('Random distribution', 'Parallel ...
        distribution', 'Antiparallel distribution')
352     name=('Plotting2_idea.Comparison_average_path_length');
353     saveas(f, name);
354 case 4
355     f=figure();

```



```

356         line1=diam(1:27);
357         line2=diam(28:54);
358         line3=diam(55:81);
359         plot(line1,'—ks','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','k','MarkerSize',10);
360         hold on
361         plot(line2,'—gs','LineWidth',2,'MarkerEdgeColor','g','MarkerFaceColor','g','MarkerSize',10);
362         hold on
363         plot(line3,'—bs','LineWidth',2,'MarkerEdgeColor','b','MarkerFaceColor','b','MarkerSize',10);
364         set(gca,'FontSize',14)
365         xlabel('Different triplets of alpha ,phi and threshold values')
366         ylabel('Network Diameter')
367         title('The influence of idea distribution on Network Diameter')
368         legend('Random distribution','Parallel ...
            distribution','Antiparallel distribution')
369         name=('Plotting2_idea.Comparison.diam');
370         saveas(f,name);
371     end
372 end
373
374 %%%%%%%%% plots for degree distribution resulting from different idea
375 %%%%%%%%% distribution
376 count=0;
377 for choice1=1:3
378     switch choice1
379         case 1
380             s1='phi=0.1';
381             m1='1';
382         case 2
383             s1='phi=0.3';
384             m1='2';
385         case 3
386             s1='phi=0.5';
387             m1='3';
388     end
389
390     for choice2=1:3
391         switch choice2
392             case 1
393                 s2='alpha=0.01';
394                 m2='1';
395             case 2
396                 s2='alpha=0.05';
397                 m2='2';
398             case 3
399                 s2='alpha=0.1';
400                 m2='3';
401         end
402
403         for choice3=1:3
404             switch choice3

```

```

405         case 1
406             s3='threshold=0.001';
407             m3='1';
408         case 2
409             s3='threshold=0.01';
410             m3='2';
411         case 3
412             s3='threshold=0.05';
413             m3='3';
414     end
415     count=count+1;
416     line1=dgr{count};
417     mine1=frq{count}/1000;
418     line2=dgr{27+count};
419     mine2=frq{27+count};
420     line3=dgr{54+count};
421     mine3=frq{54+count};
422     f=figure();
423     loglog(line1,mine1,'—ks','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','k');
424     hold on
425     plot(line2,mine2,'—rs','LineWidth',2,'MarkerEdgeColor','r','MarkerFaceColor','r');
426     hold on
427     plot(line3,mine3,'—bs','LineWidth',2,'MarkerEdgeColor','b','MarkerFaceColor','b');
428     set(gca,'FontSize',14)
429     xlabel('Log(degree)')
430     ylabel('Log(frequency)')
431     title(['The influence of idea distribution on the degree ...',
432           'distribution',' ',s1,' ',s2,' ',s3]);
433     legend('Random distribution','Parallel ...',
434           'distribution','Antiparallel distribution')
435     name=(['Plotting2_idea_comparison_degree_distribution-',m1,'-',m2,'-',m3]);
436     saveas(f,name);
437 end

```

~~~~~ I put here the matlab codes in this folder to be able to put them in the report, later I'll figure it out how remove them

## 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](http://en.wikipedia.org/wiki/Erd%C5%91s%E2%80%93R%C3%A9nyi_model).