UNIVERSITÀ DEGLI STUDI DI TORINO

Multi-agent Systems Course A.A. 2017/2018 Prof. Marco Maggiora

How the opinion variable complexity can modify the equilibrium time in a network consensus model

Federico Cinus, Federico Delussu, Nicola Pacella
https://federicocinus.github.io/SocialStock/

Abstract

In this work is introduced the consensus problem on network. In particular we use MAS to model the opinion influence between nodes in a social network and study the time necessary to reach consensus. We introduced the possibility to vary the complexity of each node opinion over a set of fields which we call *companies*. Some estimation has been done in order to make the simulation suitable for representing reality and still keep computational costs low. Results show the robustness of the proposed solution both in scale free setting and uniform degree distribution setting.

Introduction

The MultiAgent System is a powerful tool for describing behaviors emerging from human societies. For this reason it fits perfectly with the goal that we want to achieve whit this project. In particular our main goal is to describe the spreading of positive or negative opinions about companies on social networks. For this reason we implemented in our project an environment where our agents, or users, communicates with each other, exchanging opinions and influencing one another. We then analyzed the data, highlighting some important results.

1 Mas Approach

1.1 MultiAgent Systems

The development of computer science brought the necessity of moving away from machine-oriented views of programming toward concepts and metaphors that more closely reflect the way in which we ourselves understand the world. In the earlier days of computing indeed, programmer had no choice but to program through raw lines of machine code. Subsequent, programming paradigms have progressed away from such low-level views, developing from assembler languages to the more recent objects. Each development has allowed programmers to conceptualize and implement software in terms of higher level and more human-oriented abstractions.

This trend reflects the necessity of delegation, in particular in two different ways. The first one demands the systems to operate independently, without our direct intervention, while the second is the need for computer systems to be able to act in such a way as to represent our best interests while interacting with other humans or systems.Together, these necessities have led to the emergence of a new field in computer science, MultiAgent Systems.

The idea of a multiagent system is very simple. An agent is a computer system that is capable of independent action on behalf of its user or owner. In other words, an agent can figure out for itself what it needs to do in order to satisfy its design objectives, rather than having to be told explicitly what to do at any given moment. A MultiAgent System is one that consists of a number of agents, which interact with one another, typically by exchanging messages through some computer network infrastructure. In the most general case, the agents in a MultiAgent System will be representing or acting on behalf of users or owners with very different goals and motivations. In order to successfully interact, these agents will thus require the ability to cooperate, coordinate, and negotiate with each other, in much the same way that we cooperate, coordinate, and negotiate with other people in our everyday lives.

It is important to stress that each agent can fulfill his aims only in a specific environment, so that agents and environment have the same importance and are strictly related to each other. This link is so strong that from a single agent point of view, other agents are embodied in the environment itself.

The final aim of the MultiAgent System is to observe and study the emergence of a collective intelligence derived from the interaction of the agents.

1.2 Mas for opinion spreading

The MultiAgent System approach fits perfectly for our main goals. In particular, we want to reproduce the exchange of opinion that happens between users of a generic social network in regard of some companies. In our model, the users will correspond to the agents, and they will aim to communicate with other agents with the purpose of exchanging opinion on some companies, implemented as purely reactive agents. The ambient instead will be a social network in which agents, or users, will find other agents that share a similar opinion on the companies.

1.3 Jade

JADE (Java Agent DEvelopment Framework) is a software Framework fully implemented in the Java language. It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA specifications and through a set of graphical tools that supports the debugging and deployment phases. A JADE-based system can be distributed across machines (which not even need to share the same OS) and the configuration can be controlled via a remote GUI. Besides the agent abstraction, JADE provides a simple yet powerful task execution and composition model, peer to peer agent communication based on the asynchronous message passing paradigm, a yellow pages service supporting publish subscribe discovery mechanism and many other advanced features that facilitates the development of a distributed system.

2 The Model

The simulation is made up by a series of D temporal steps which we call days. On first day the Users' opinions are randomly initialized. On each day the Users influence each other with pairwise interactions in which opinions can be exchanged.

2.1 Initialization

The initial world's configuration is given by the instantiation of N Users. The Users live in a world with C companies, which are indexed from 0 to $C-1$, and are provided with a set of opinions, one for each company.

An opinion is a positive integer which ranges from 0 to $R-1$, let R denote the opinion range. For each User the opinions are stored within a vector $\mathbf{v} \in \mathbb{N}^C$ which we denote as the opinion vector, the entry $\mathbf{v}_c$ is the opinion on the c-th company.

Each User has an inclination $I \in \{-1, 0, 1\}$ whose possible values denote respectively a "bad", "neutral" and "good" averaging opinions across the companies [more details in the appendix].

Each User has a degree k which is the number of Users to whom he can send messages during the day.

Parameter k is generated randomly according to the scale-free distribution with parameter γ and is never changed during the simulation; if γ is set to zero then the degree distribution is uniform.

Algorithm 1: Init$\{N, C, R, \gamma\}$

```
N ;                                              // Number of Users
C ;                                              // Number of Companies
R ;                                              // Opinion Range
γ ;                                              // Scale-Free parameter

U = ∅
for n ← 0 to N − 1 do
    k_n = Sample(γ) ;                            // sample from power-law distribution
    v_n = 0_C ;                                  // C-dimensional Opinion Vector
    for c ← 0 to C − 1 do
        v_n[c] = U{0, R − 1} ;                   // sampling entries from pmf U{0, R − 1}
    end
    I_n := Incl(v_n) ;                           // Computing the Inclination
    u_n = (k_n, v_n, I_n) ;                      // n-th User instantiation
    U ← U ∪ {u_n}
end
return U
```

2.2 Interaction

On each day Users advertize their Inclination and Degree by registering to the Directory Facilitatory (DF); the DF is implemented by Jade and could be compared to the "Yellow Pages" phone book. Once all the Users have registered to the DF they can make queries in order to look for other Users sharing the same Inclination and then collecting them in a list whose length is equal to their degree. Then communication is accomplished through the exchange of messages to Users contained in the list. The messages are implemented by the JADE class ACLMessage and their respective performative is REQUEST.

Each time a User receives a message he interacts with the sender.

Let's consider an interacting couple and call the two Users A and B having respectively degree k_A and k_B, the interaction consists in these steps :

1 Opinion comparison

B compares its opinion vector $\mathbf{v}_B$ with $\mathbf{v}_A$, given the subset of companies on which the two Users' opinion are different, a company's index c is extracted according to a uniform distribution. If $\mathbf{v}_A$ and $\mathbf{v}_B$ are equal the interaction stops.

2 Influence

- The opinion on c-th company remains unchanged for B while A changes it to B's opinion. This event occurs with probability $k_B/(k_A + k_B)$.
- The opinion on c-th company remains unchanged for A while B changes it to A's opinion. This event occurs with probability $k_A/(k_A + k_B)$.

Algorithm 2: Interaction$\{u_A, u_B\}$

v_A, v_B ; // Opinion Vectors of u_A and u_B
k_A, k_B ; // Degrees of Users of u_A and u_B

$\mathcal{I}_D = \{c : v_A[c] \neq v_B[c]\}$; // indexes of differing opinion
$c_D = Sample(\mathcal{I}_D)$; // single sample from $\mathcal{I}_D$

$u = \mathcal{U}[0, 1]$; // sample from pdf $\mathcal{U}[0, 1]$
$p_A = \frac{k_A}{k_A + k_B}$; // prob A influences B

if $u \leq p_A$ **then**
  $v_B[c_D] \leftarrow v_A[c_D]$; // B attains A's opinion on company c_D
else
  $v_A[c_D] \leftarrow v_B[c_D]$; // A attains B's opinion on company c_D
end

3 Data Analysis

This section wants to deal with the following objectives:

1. visualize the social network for each day of the simulation
2. find equilibrium time in IncDistr vs time plot
3. find equilibrium time in opinions vs time plot

We underline that in order to represent a complex reality we deal with simulations with more than one company. Moreover we did some estimations to make simulation suitable for computational costs; in particular we fixed the number of companies to 3 bringing to equilibrium times included in 100 time steps ranges, and the number of nodes to 100. Since the only effect of incrementing those parameters is pushing up in time the equilibrium time, it is reasonable to not vary them.

3.1 Social network evolution

The following plot shows how inclination evolves towards consensus during simulation.

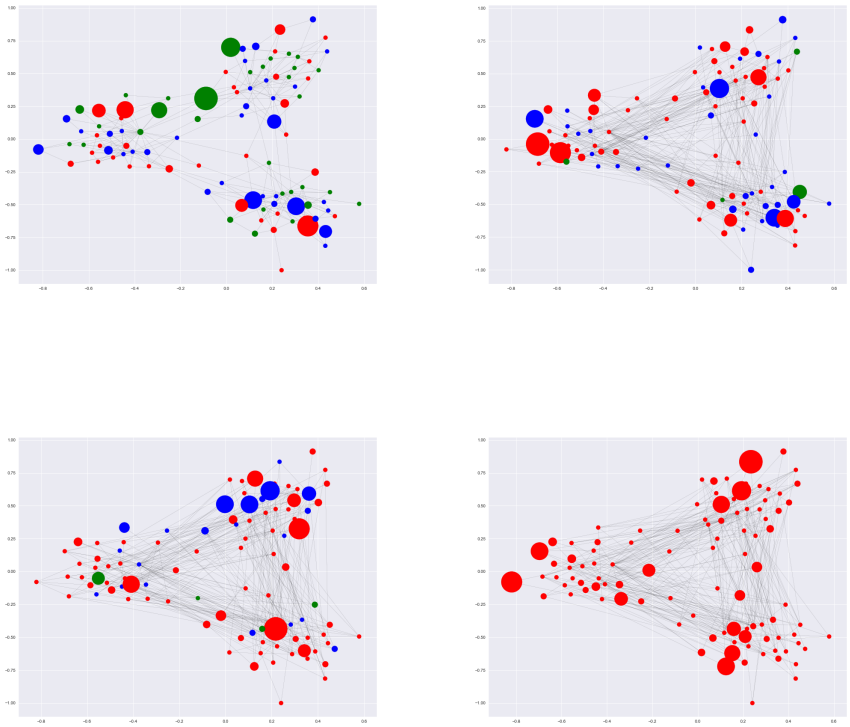


Figure 1: Network evolution

3.2 Equilibrium time for IncDistr variable

We define equilibrium time the particular time step in which the dependent variable reaches a plateau. Considering the *IncDistr* variable as the dependent one, in a *IncDistr* vs *time* plot we can find the equilibrium time described above. The following plots show particular simulations with different parameters and their equilibrium time.

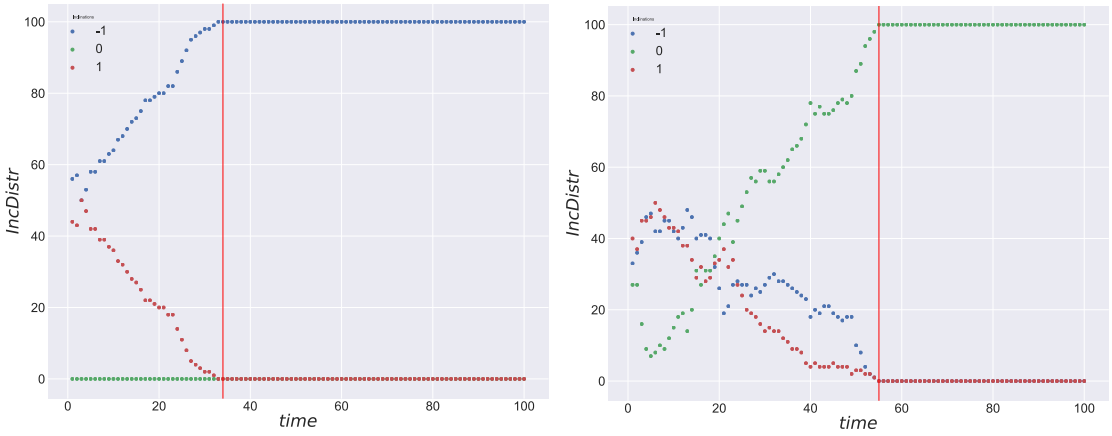


Figure 2: IncDistr vs time for 3 companies, opinion range 2 (left) and 20 (right), scale free graph

Since the range of opinions can improve the variety of opinions indeed, so that the complexity of the system goes up, we can see that equilibrium time grows with *Opinion-Range*. Next paragraph will provide a more detailed description giving a confront with the global opinion variable for each company (which is the opinions' sum over all nodes). Moreover, considering the network degree distribution, we can find that a uniform distribution can achieve more communication with respect to power law one. Since nodes have on average higher degree they can influence each other more efficiently causing lower equilibrium time measures.

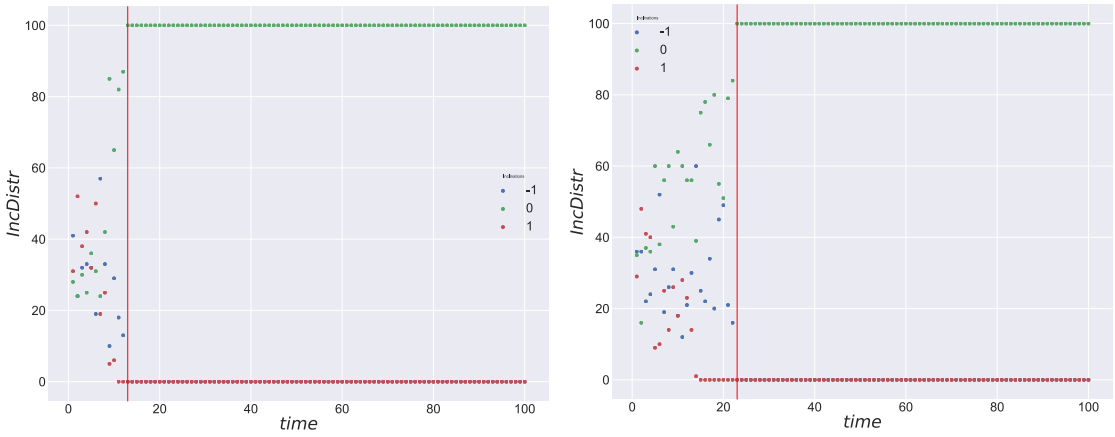


Figure 3: IncDistr vs time for 3 companies, opinion range 2 (left) and 20 (right), random uniform graph

3.3 Equilibrium time for opinions

In the same way as the previous section we can plot the sum over nodes of all opinions for each company. In this way we can see how this quantity varies in time, in order to find equilibrium time. We report again the *IncDistr* plots for a confronting aim.

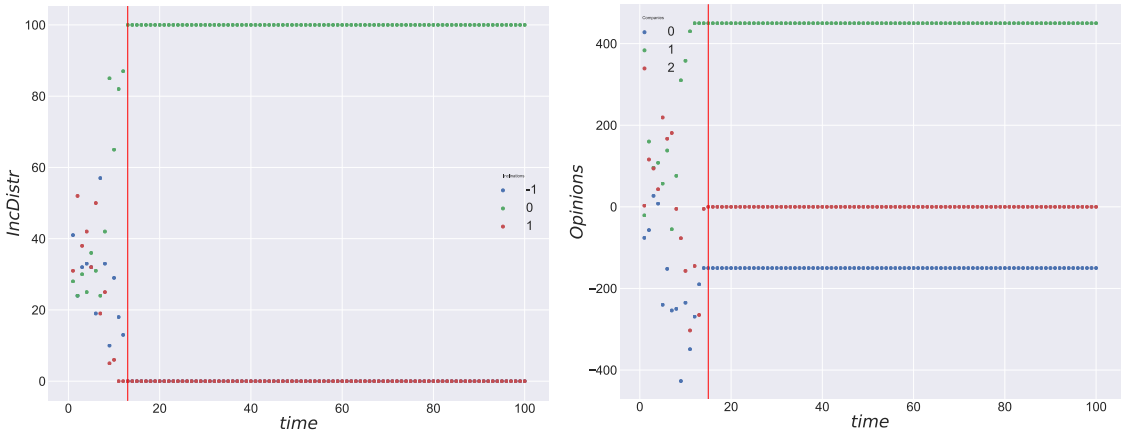


Figure 4: IncDistr and opinions sum vs time for 3 companies, opinion range 2, random uniform graph

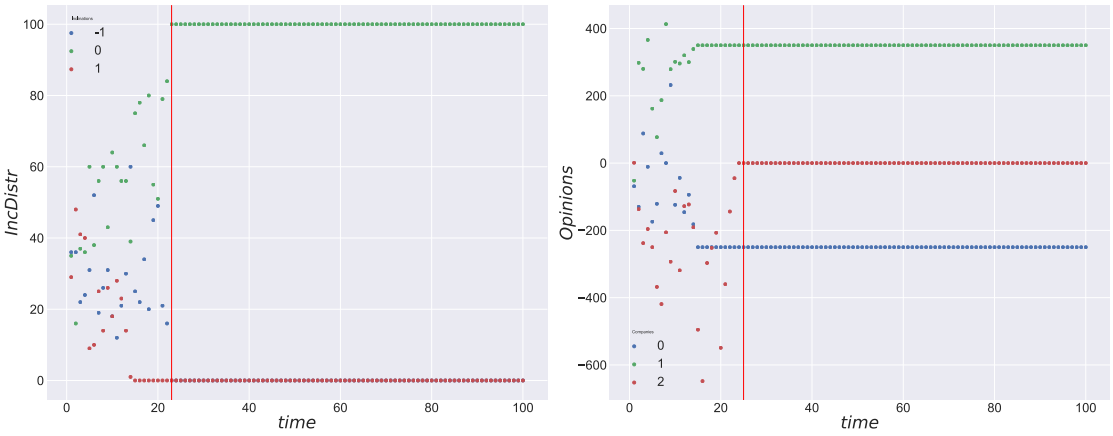


Figure 5: IncDistr and opinions sum vs time for 3 companies, opinion range 20, random uniform graph

It is deeply interesting to notice that the dependent variable's complexity can achieve in influencing the equilibrium time. In particular since *IncDistr* is derived from opinions using its mean, we can say that it has a lower complexity than the opinions themselves. As we can see equilibrium time in opinions plot is higher than the *IncDistr*'s one.

We report the following statistics for a scale free network with $\gamma = 2.5$:

Equilibirum Time	OpinionRange=2	OpinionRange=10	OpinionRange=20
IncDistr	36 ± 17	50 ± 22	57 ± 44
Sum Opinions	40 ± 18	56 ± 23	74 ± 42

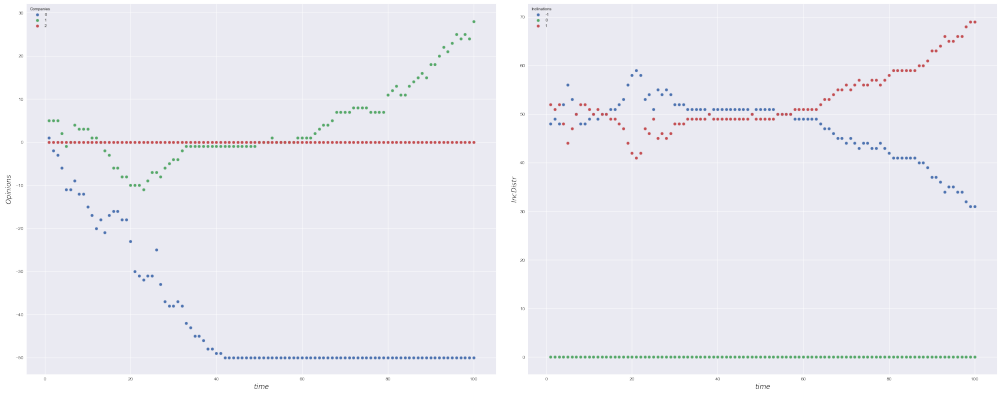
Table 1: Equilibrium time for opinions over 3 companies in a 100 nodes scale free network ($\gamma = 2.5$)

The range of opinions can push up the equilibrium time, moreover it is higher in the sum of opinions plots. Notice that also standard deviation increases as *OpinionRange* increases too. Some statistics have been collected in order to show the lower values of the equilibrium time in case of uniform degree distribution of the network:

Equilibirum Time	OpinionRange=2	OpinionRange=10	OpinionRange=20
IncDistr	15 ± 4	15 ± 6	18 ± 8
Sum Opinions	17 ± 2	17 ± 6	19 ± 8

Table 2: Equilibrium time for opinions over 3 companies in a 100 nodes random network (uniform degree distribution)

Given the model stochasticity we stress out that equilibrium time measures are not always present inside 150 time steps range. The following plot is an example.



Conclusions

In conclusion we verified that as the *OpinionRange* increases the equilibrium time increases too. This is a consequence of a higher complexity of the users' opinions. We also found that scale free setting has a higher equilibrium time measure with respect to the uniform degree distribution setting. In fact the hub presence, combined with the fact that low-degree nodes can still achieve its influence, produces a more dynamic network. Moreover the standard deviation is dependent both on the *OpinionRange* and the network model setting. The model complexity is therefore the determinant factor for equilibrium time in network consensus.

Appendix

Inclination computation

The Inclination for each User is computed by considering the average of the opinions across the companies.

Let's consider an opinion vector $\mathbf{v}$ with entries $\mathbf{v}_c$ ($c = 1, .., C$). Each entry $\mathbf{v}_c$ is drawn uniformly from the discrete set $\{0, 1, .., R-1\}$. Recalling that the uniform discrete distribution $\mathcal{U}\{a, b\}$ has mean $\frac{a+b}{2}$ and variance $\frac{(b-a+1)^2-1}{12}$ we have that $\mathcal{U}\{0, R-1\}$ has mean $\frac{R-1}{2}$ and variance $\frac{R^2-1}{12}$

Let S be the average of $\mathbf{v}$ entries:

$$S(\mathbf{v}) = \sum_{c=1}^{C} \frac{\mathbf{v}_c}{C}$$

For the central limit theorem, for $C \to \infty$ the random variable S will follow a normal distribution with mean $\frac{R-1}{2}$ and variance $\frac{R^2-1}{12C}$.

In order to assess wether the inclination $I = -1$ we check if the value of S lies under the quantile of order $1/3$ of the normal distribution $\mathcal{N}(\frac{R-1}{2}, \frac{R^2-1}{12C})$. If $I = 0$ then S is between the $1/3$ and $2/3$ quantile, and $I = 1$ if S is over the $2/3$ quantile.

The $1/3$ quantile of the standard normal $\mathcal{N}(0, 1)$ is $q^{(S)}_{1/3} = -0.4399132$ and by simmetry $q^{(S)}_{2/3} = -q^{(S)}_{1/3}$. The $1/3$ quantile of $\mathcal{N}(\mu, \sigma^2)$ is $q_{1/3} = \mu + \sigma q^{(S)}_{1/3}$ and the same applies for the $2/3$ quantile.

So, in summary, given the User's opinion vector $\mathbf{v}$ we compute its inclination according to the average of components S:

$$I(\mathbf{v}) := \begin{cases} -1 & \text{if } S < \frac{R-1}{2} + q^{(S)}_{1/3}\frac{R^2-1}{12C}, \\ 1 & \text{if } S > \frac{R-1}{2} + q^{(S)}_{2/3}\frac{R^2-1}{12C}, \\ 0 & \text{if else} \end{cases}$$