Study of Social Contagion Processes In a Music Based Social Network

Nicola Mularoni - 2523003
Bojan Simoski - 2524079

Abstract

This report should provide all the necessary information so that you, the reader, can have a complete vision of our ideas for this project. The task was not easy, but we hope our thoughts are clear and straightforward. The purpose of this project is to summarize the knowledge gained during this course, in which, we learned a lot of different aspects of modeling agents, their internal states or the networks they belong to. Dealing with such a broad course, gives a chance of really effective brainstorming, at least that was our case. But not all of the ideas that have crossed our mind, can be effectively modeled in practice, so after considering all possible solutions, we decided to work on a dynamic network modeling. Why this specific case? We wanted to investigate how the Social Contagion Process works for one of the most subjective opinion, the music taste.

The general questions about this are:

- How does the connection between the users evolves with respect to their musical taste?
- Is there the formation of clusters of users based on the same taste?
- Does the difference of musical taste of the users affect the shape of the network?

Last.fm

In order to give an answer to the question just presented, we decided to use the website Last.fm which is, as they says, a music recommendation service - one of the most popular sites about music which allows the user discovering more music similar to its taste.

Each user has his own music world here: events he attends to, favorite musicians, friends, neighbors (people who are not friends but have most similar music taste)…

Our goal was to choose a user with a proper activity on the website and with a number of friends suitable for our experiments, in order to form a network based on them and analyze how their music preference for a particular music genre changes over time, as a result of the connections between the agents.

The Model

The idea is to define a social contagion model since a lot of behavior phenomena have been found to spread through agents of social networks in a contagious manner.

Considering the nature of the topic we have chosen, we thought to adopt a dynamic network based on a homophily principle which states, in a totally different manner from the famous quote that opposites attract, the more you're alike, the more you like. Based on this, the connection strengths of our network are getting stronger the more the music taste between agents is similar. We decided to choose four different music styles and build opinion of the agents about them: rock, jazz, pop and electronic music.

The general idea of the model we used is already developed and more information can be found in the literature [1].

First of all we show how the states for different opinion *s* of agent *i* develop over time:

$$q_{s,i}(t+\Delta t) = q_{s,i}(t) + \eta_i \sum_{j \neq i} \gamma_{j,i}(t)(q_{s,j}(t) - q_{s,i}(t))\,\Delta t \qquad (1)$$

We are taking into account the difference between each two agent's opinions and then multiplying with the connection strength between those two agents. The initial connection strengths γ_{ij} are set based on this equation:

$$\gamma_{ji=}\varepsilon_j \alpha_{ji} \delta_i \quad (2)$$

The idea is to make the model more complex, including the internal impression and expression processes occurring at each agent so that one can analyze the different roles they have in the network. Here ε_j is the expressiveness factor for the expression process, α_{ji} is the channel strength between agent j and agent i for the state transfer and δ_i is the openness factor for the impression process.

The weights of this network indicate how much one agent influences another one and is dependable of the function $f_{i,j}$, which actually provides the homophily principle by taking into account the differences of the opinions of two agents-the smaller this difference is the bigger value for the influence is gained. This function is designed so that the case $0 \leq \gamma_{i,j} \leq 1$ it's always satisfied.

$$\gamma_{i,j}(t+\Delta t) = \gamma_{i,j}(t) + f_{i,j}(\gamma_{i,j}(t), q_{s,j}(t), q_{s,i}(t)) \quad (3)$$

$$f_{i,j}(X,Y,Z) = X\alpha_{ij}(\beta_{ij} - (Y-Z)^2)(1-X) \quad (4)$$

Here α_{ij} is the speed parameter and β_{ij} is the threshold parameter. These parameters influence the behavior of the agents, as it will be shown in the simulation.

Data Gathering

Another key factor for which we chose Last.fm is its richness in terms of API[1], essential for us in the task of collecting the needed data for our model. We have collected the agents states with the API for 20 days, from the 15th December 2012 to the 3rd of January 2013.

This was quite laborious work, since we need a lot of data (worked with 96 agents) that requires a lot of time to be gained together with the fact that as every Social Network which expose its API, Last.fm has a limit in the request you can perform in a certain period of time.

[1] For more information about the API provided by Last.fm visit: http://www.lastfm.it/api

The next paragraphs are intended to explain more in detail this part of our project.

Opinion States

In order to get the initial opinion states of the agents, we are taking **all** the artists that each user has in his favorite list and then determine the genre that the artist belongs to. All of this information are available in a XML file which is the answer of the Last.fm server to every API request.

```
artists = User.getTopArtist
for each artist in artists
    tags = artist. getTags
```

Figure 1: Pseudocode for the computation of the initial opinion state of one agent

We are counting the number of rock, pop, jazz and electronic artists using their tags. Since a musician can belong to more than one genre we also keep track about it, aware about not gathering redundant information. At the end after counting everything we simply make these calculations:

```
$opinionRock=$nRocks/$nRelated
$opinionJazz=$nJazz/$nRelated
$opinionPop=$nPop/$nRelated
$opinionElectronic=$nElectronic/$nRelated
```

We thought that in order to have significant values for the agent's states, the minimum period of time before the next request, was one day so the other states we collected for the model validation are obtained in the same way but the request is limited to the **last day**.

Expressiveness

For the expressiveness factor we decided that a good indicator may be the number recently listened tracks (available from the API) for each agent for the last 30 days, and then in order to create the appropriate ratio and to keep the ε_j parameter in range [0,1] we calculated the expressiveness of each agent by dividing his number of listened tracks with the maximum value, the biggest number of listened tracks, comparing all agents.

```
for each user in users
    tracks = User. getRecentTracks (one month)
```

Figure 2: Pseudocode for the computation of the expressiveness

Channel Strength

For the channel strength α_{ji} between each of the agents in our network, we used an API function called TasteoMeter which compares the tastes of each two users using a list of shared artists and the number of common friends between them in an equation which is not public.

```
for each user_i in users
    for each user_j in users
        score = Tasteometer.compare(i,j)
```

Figure 3: Pseudocode for the computation of the channel strength

Openness

Since the openness in our case demonstrates how much an agent is open-minded about different kind of music, we figured out this solution to present that in a mathematical manner: for each agent first we find what kind of music is he keen about or in other words the max of all of his opinion state values. Than we calculate the difference between each state value and this max value, and compare this differences with a threshold of 0.2. The idea is that an agent who is more open to new music will sure have small difference values (hopefully smaller than 0.2) - meaning his initial opinion states about rock, jazz, pop and electronic music have pretty similar values. Next we count how much of these differences are smaller than the threshold and based on this number generate a value for the openness factor.

```
for each user_j in users
    s_Max=compute max state between states
    for each state_i in states
        if 0<diff(s_Max - s_i)  && diff(s_Max - s_i)<0.2
            threshold_count++
    if threshold_count==0
        openness(user_j) = random number between (0 – 0.5]
    if threshold_count==1
        openness(user_j) = random number between (0.5 – 0.7]
    if threshold_count==2
        openness(user_j) = random number between (0.7 – 0.9]
    if threshold_count==3
        openness(user_j) = 1;
```

Figure 4: Pseudocode for the computation of the openness

Model Simulation

After gathering the needed data and transform it into appropriate form for Matlab, it was time to run some simulations and see the behavior of our network. We tried to run some simulations in order to see the specifics of the network and the behavior of the model more accurately. So for each of the four states, we had 6 simulations giving different values for the alpha and beta parameters, as those are the one that influence the most the behavior of the model. The number of time steps was set to 40, with $\Delta t=0.01$. The value of the eta parameter was set as a random number between 0.5 and 1 for each agent.

From the pictures below, where we represent change of the opinion state values of all the agents over time, one can conclude that bigger the value of the speed parameter (alpha) and smaller the value of the threshold parameter (beta), the more clusters are formed. This pattern is present at all of the experiments we did. Clusters in the network occur as the result of converging values of the opinion states in a group of agents considering some kind of music. This situation is expected as we are working with social contagion process based on homophily principle, so as the weights on the connections between agents get stronger (bigger), the agents opinion states start to converge. Also the opposite case is detected: as the connection strength between agents weakens they converges on different values (clusters). As the state values for jazz music follow really similar values (low for almost all agents) the formation of multiple clusters here is impossible even with big values for alpha and small for beta.

The longer the simulation lasts, the less the number of clusters is. We can compare the following cases- analyzing the rock opinions with alpha=15 and beta=0.01, with simulation lasting 40 and 60 time points respectively:

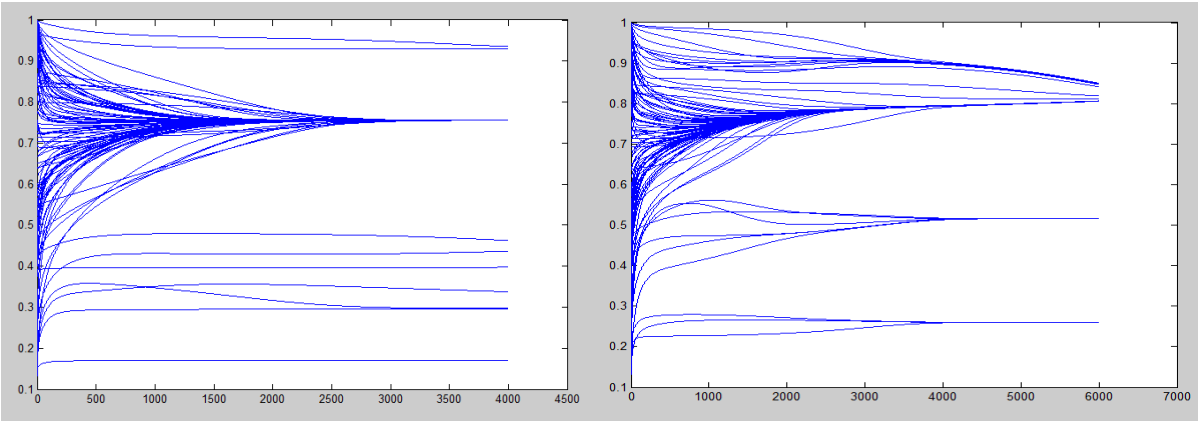
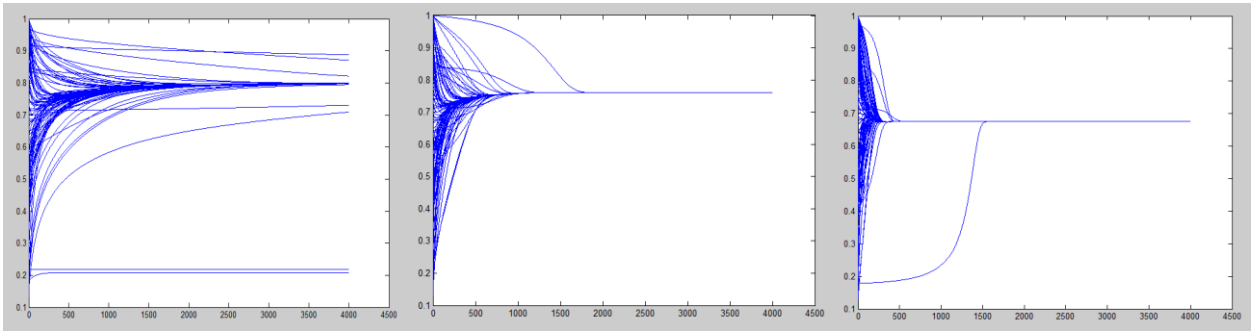


Figure 5: Comparison between simulations lasting 40 time points and 60 time points with the same values of alpha and beta

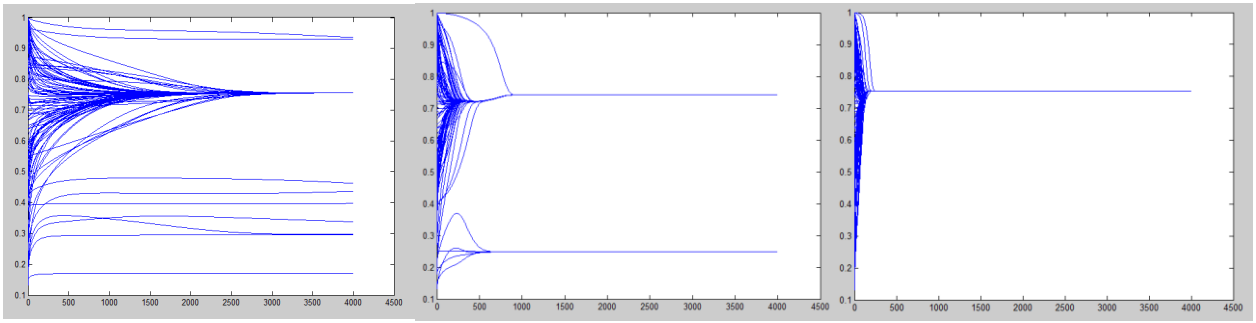
At the second plot the states of agents are still converging and we can expect that choosing more time points will result in even smaller number of clusters.

We can see that the group of people who formed our network has really big preference for rock, pop and electro music, while jazz is not the most popular choice. This analysis would be completely different if we started our network building from a user that has Miles Davis tracks as his most played… Still our starting point was the user called *Liverald* who has big preferences on rock music (based on the initial opinion states we gained) so it's not surprising that most of her friends on last.fm actually are listening similar music to some degree.

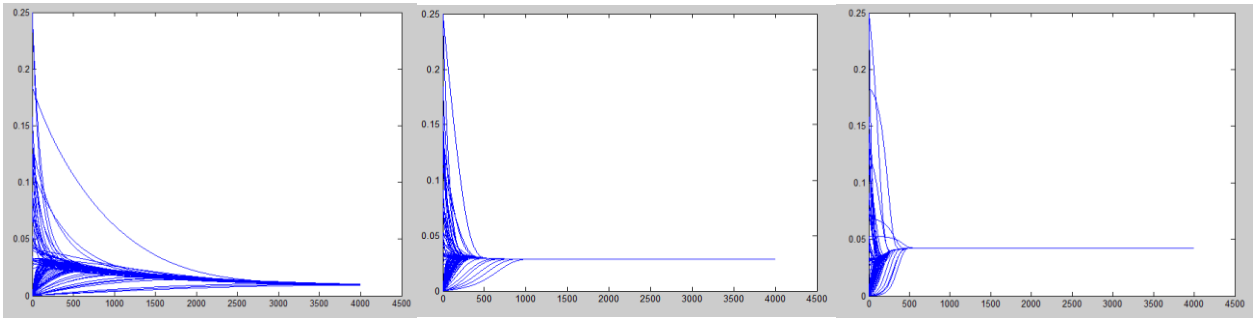
Rock Opinion States: alpha=5 beta=0.01; 0.1; 0.3; respectively



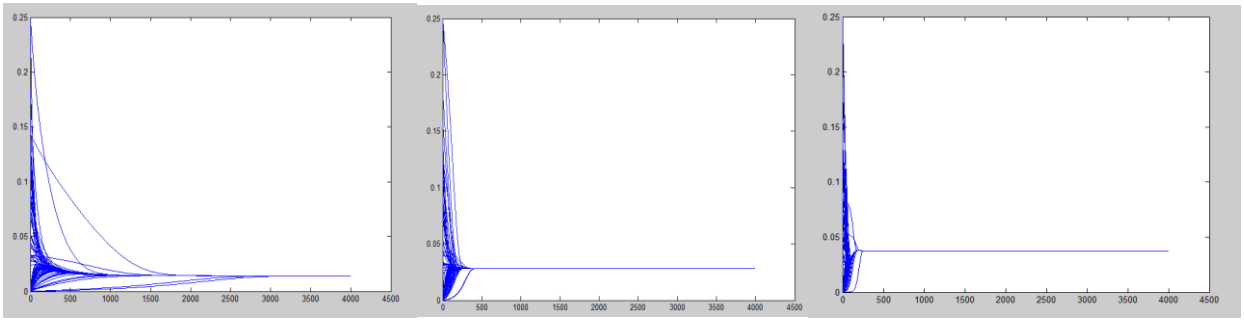
Rock Opinion States: alpha=15 beta=0.01; 0.1; 0.3; respectively



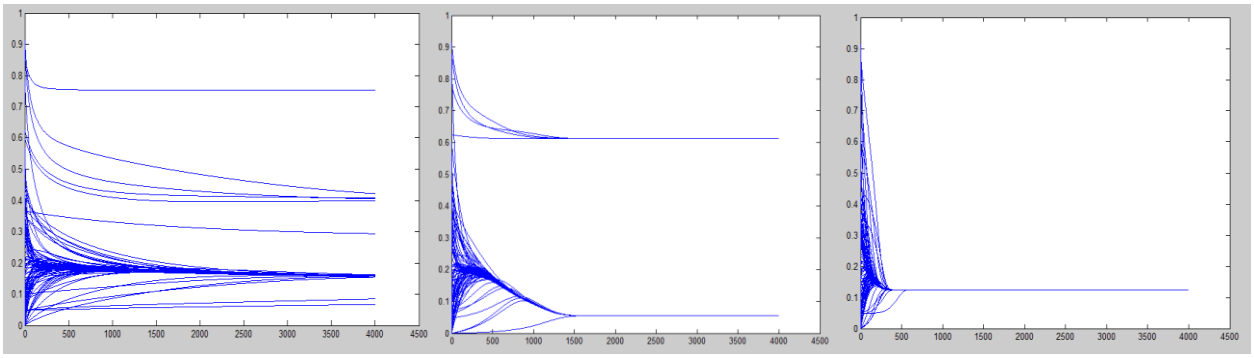
Jazz Opinion States: alpha=5 beta=0.01; 0.1; 0.3; respectively



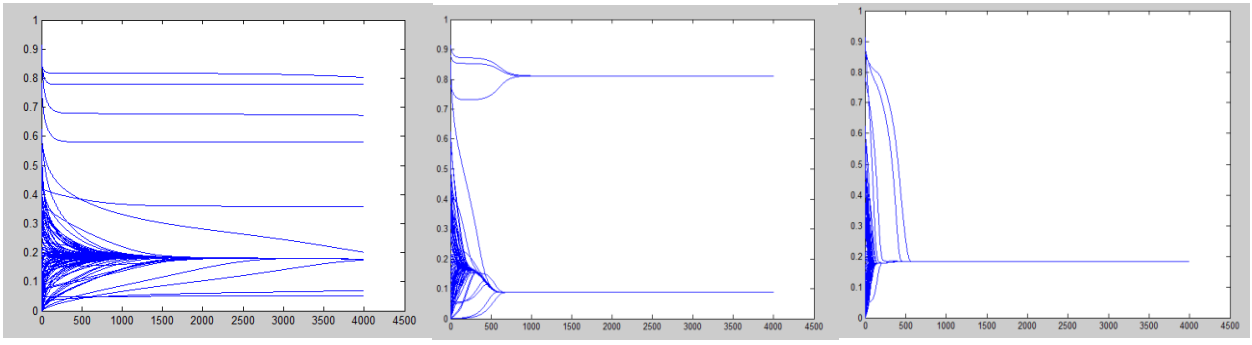
Jazz Opinion States: alpha=15 beta=0.01; 0.1; 0.3; respectively



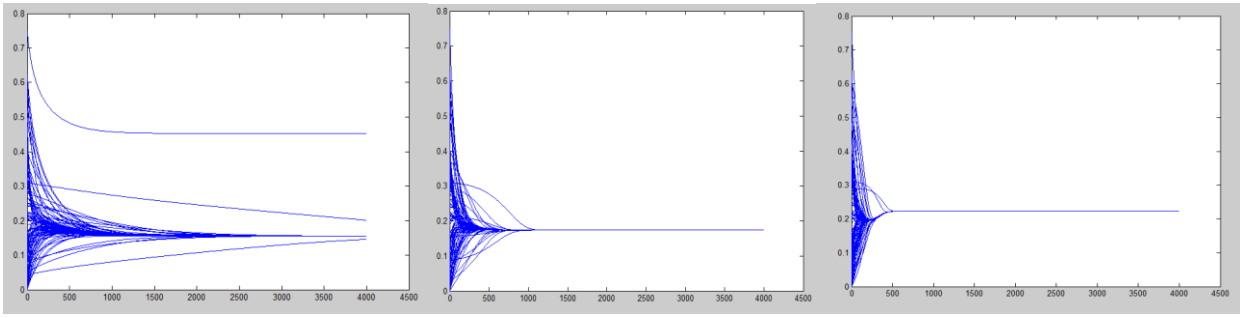
Pop Opinion States: alpha=5 beta=0.01; 0.1; 0.3; respectively



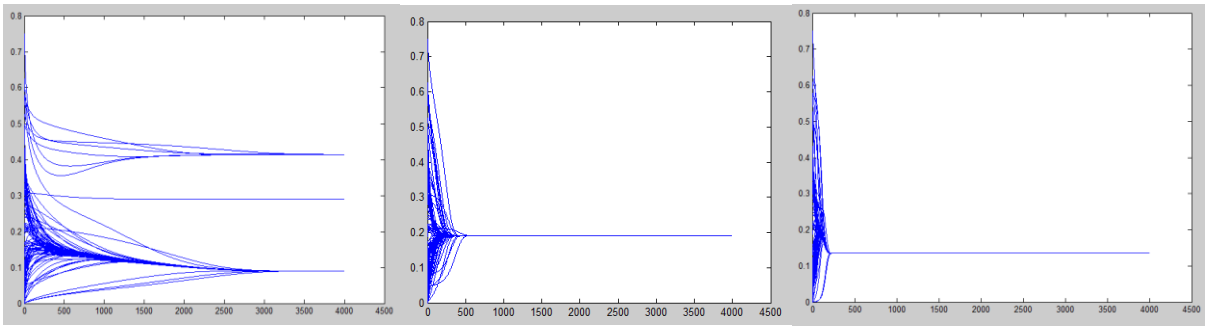
Pop Opinion States: alpha=15 beta=0.01; 0.1; 0.3; respectively



Electronic Opinion States: alpha=5 beta=0.01; 0.1; 0.3; respectively



Electronic Opinion States: alpha=15 beta=0.01; 0.1; 0.3; respectively



It can also be seen that with high values of beta the convergence in cluster is too fast and it does not reflect the real behavior of the network we are taking into account; in fact with real agents (quite active on the music platform) we have noticed a slower change in the user's opinion also because in our specific case a change in the musical opinion means to listen and to enjoy some artists, which is a process that requires time.

Due to this reason we have modified the parameter space of beta [0.01, 0.15] with respect to the original implementation of [1] since the real behavior of the agent's states is not compatible with the range of values (0.15, 0.4].
The range for the parameters is summarized in Table 1:

η_i	α_{ij}	β_{ij}	$\gamma_{i,i}(0)$	$q_{s,i}(0)$
[0.5, 1]	[1, 20]	[0.01, 0.15]	[0, 1]	[0, 1]

Table 1: Acceptable ranges of the parameters of the model

Network Analysis

A big part of our project is concentrated on building the network of the agents we are investigating. Using the knowledge from working with Gephi we analyzed the little social network we have here.

The network has been extracted with the API (basically with *User.getFriend*s and *User.getInfo*), building during the extraction the corresponding GEFX file with the XML Schema specification 1.2 readable with Gephi. As already mentioned, the node from which we started is the user called *Liverald*, gathering her friends and their connections; then we have limited the network (and consequently the analysis) to the connections between the users without introducing new users (connected with *Liverald* friends).

The network made by *Liverald*'s friend and the friends of this friend (so our supra network) however was a Scale Free network since how it can be seen from Figure 6 undergoes to the Power Law Distribution.

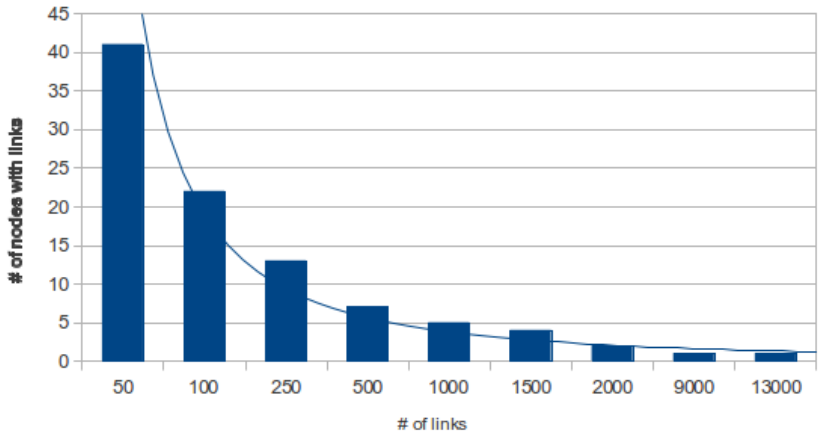


Figure 6: Distribution of node degree of our supra network compared with the Power Law Distribution

Our network consist of 96 agents and it's an undirected graph (since the friendship is mutual) consisting of 922 edges, where nodes (agents) are connected with edge if there exist friendship between two agents. As a starting point we performed a Force Atlas algorithm so that we can layout our graph, to make it more representative. The average degree of our network has a value of 19.208, with *Liverald* being the node with most connection (not surprisingly); Hermes3megisto is the second node with most connection since it was the biggest in the supra net with almost 13.000 links so we can also hypnotize that those 2 are the most influent agents in our network for what concerns the contagion process.

Only 8 of the 96 agents have more than 50 connections, while 36 of them have 10 or less. This is an indicator that also our network is a scale-free network. In order to prove this network property we run

an open-source script in Matlab, plfit.m, that calculates the maximum likelihood estimate of the scaling exponent – the gamma value. In our case we got γ=2.3994, that is a good indicator of the scale-free network.

Next we calculated the average path length a parameter that gives information about how close are nodes from each other. The value of the length is 1.89. One definition says that small-world network is defined if the typical distance L between two randomly chosen nodes grows proportionally to the logarithm of the number of nodes N in the network. In our case we have $1.89 \approx \log_{10} 96 \equiv 1.89 \approx 1.98$. This is not a surprise as a lot of real world social networks actually have small-world characteristics. This calculation in Gephi also gives us information about the centrality of nodes an indicator of the influence of a node in the group. We have chosen to analyze the betweeness centrality of the nodes, that will determine the size of the nodes in the graph. The result of this analysis is pictured in

Figure **7**.

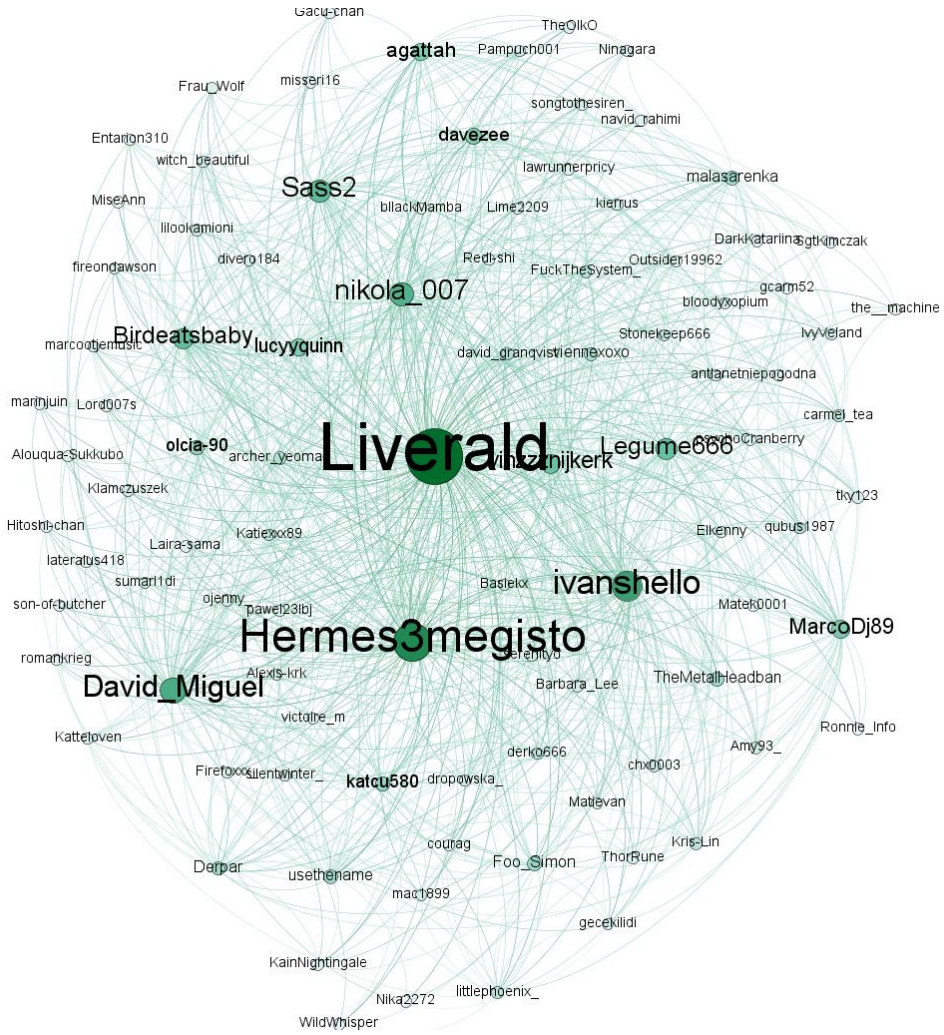


Figure 7: Outcome of the Network Analysis

Another thing that we investigated is the formation of clusters in the network, or as they say in Gephi, community detection. Running the algorithm gave us 4 clusters with 31.25% of the agents belonging to the dominant group, followed by 28.12%, 28.12% and the smallest one of 12.5%.

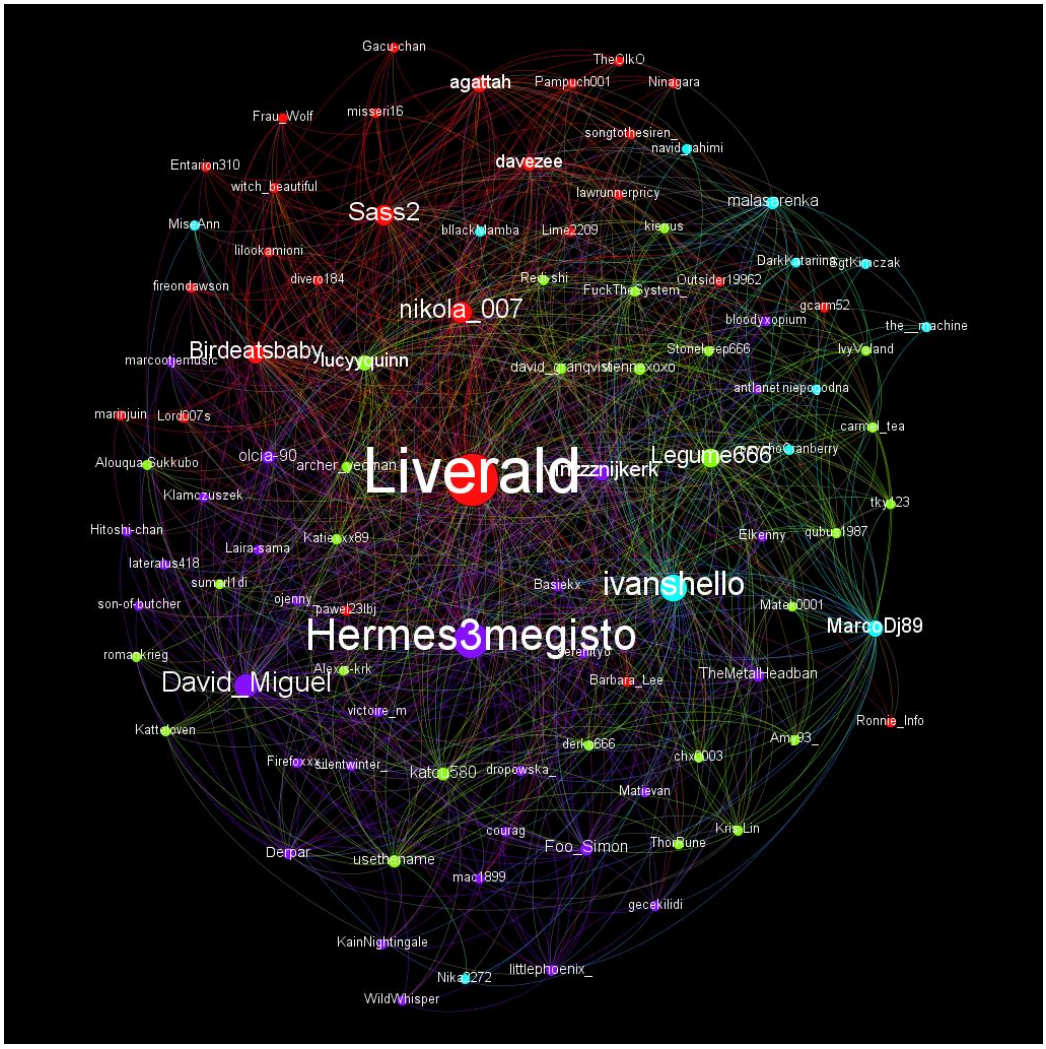


Figure 8: Clusters in the network

Model Validation

In order to perform the validation of our model we have chosen to use Linear Search and Simulated Annealing; to implement this two search techniques we started from the *tuneparameters.m* file present in the Assignment 4 made during the course, adapting it for our model.
Since the data for the agent's states was collected every day, we decided to keep a time interval of 15 minutes in order to be consistent with the Δt used in the simulations (1/100 of 1 day = 15 minutes).

The parameter estimation has been performed for the parameters α and β which, as it can be found in [1], are the two most sensitive parameters.
Considering that the number of parameters is 2 one could think that Linear Search is more suitable as search method but with the granularity of 1 for α and 0.01 for β we obtain 3000 computational steps to perform in order to explore the entire space of the parameters; on the contrary with a value of 1e-10 for *Tmin*, with the Simulated Annealing we obtain a value of 2092 computational step which is independent from the number of parameters; this value can be obtained with the equation (5) and (6) with an alpha value of 0.99.

$$j_{Max} = \left\lceil \frac{\log(T_{min})}{\log\left(cool(T_{init})\right)} \right\rceil \quad (5)$$

$$cool(T) = alpha * T \quad (6)$$

	Time	**Computation Step**
Linear Search	≈ 10 hours and 35 minutes	20*150 = 3000
Simulated Annealing	≈ 7 hours and 23 minutes	2092

The time needed for the two search methods is based on simulation of 30 time points and a desktop computer equipped with an Intel i7 Quad-Core 3.5gHz and 8GB of RAM.

Linear Search Results

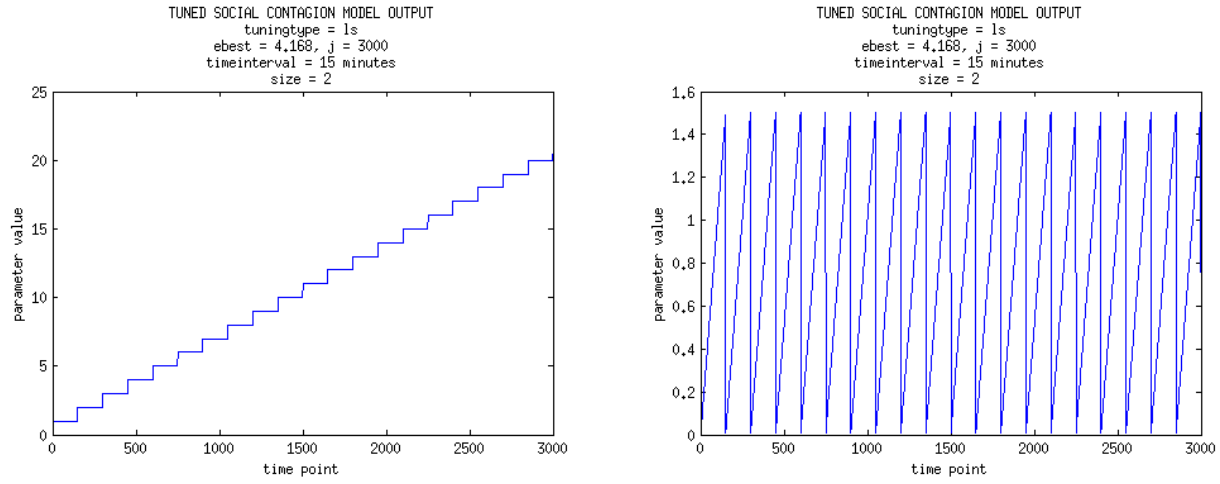


Figure 9: Respectively the parameters' alpha and beta values over time

As it can be seen from Figure 9, we have decided to try all the values for beta, starting from every value of alpha, which means that the value for alpha is changed every 150 time points while the value for beta starts again from 0.01 every 150 time points.

The energy level starts from a high value of 21 and starts dropping when the value for the parameter alpha reaches 13 and then drops again as the value reaches 17 which is the value for this parameter that performs the best validation.

The value identified for beta at the end of the search is 0.01, the beginning of the range for this parameter.

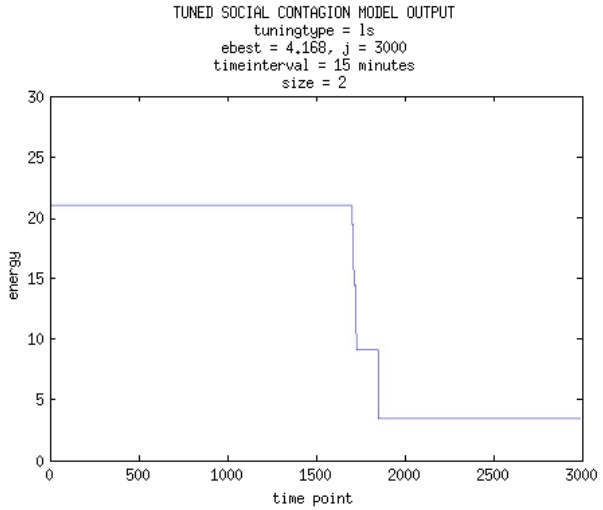


Figure 10: Best energy values evolution during the Linear Search

The green marker in the following Figures, corresponds to the real state of each agent, obtained with the API, our validation set is so composed by 96 times 20 time points.

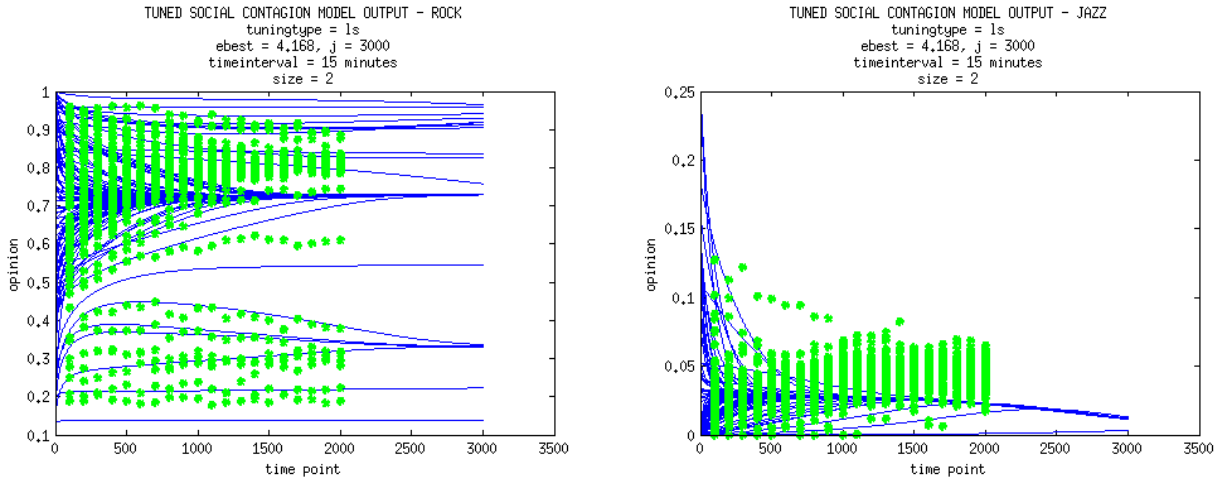


Figure 11: Output with the best parameter found with the Linear Search method with the marker for the real agents' states respectively for Rock and Jazz opinions

While for Rock, Pop and Electronic opinion states we have obtained a quite good validation of the model, for the Jazz opinion state since the initial opinion are really low and probably there is noise in the data gathered, the approximation is really low; this is also the reason why we have obtained a good energy value but quite high anyway.

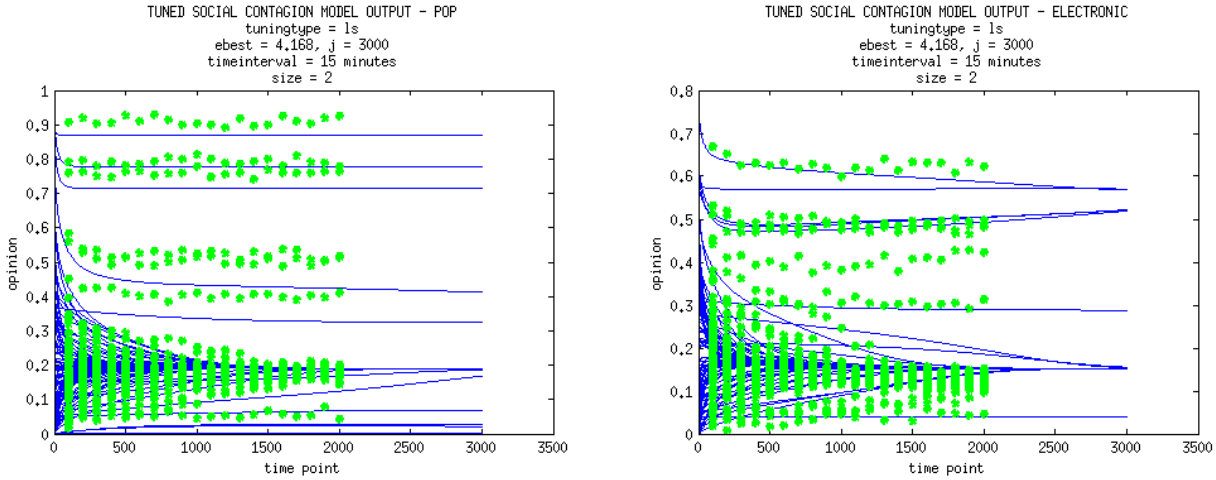


Figure 12: Output with the best parameter found with the Linear Search method with the marker for the real agents' states respectively for Pop and Electronic opinions

Simulated Annealing Results

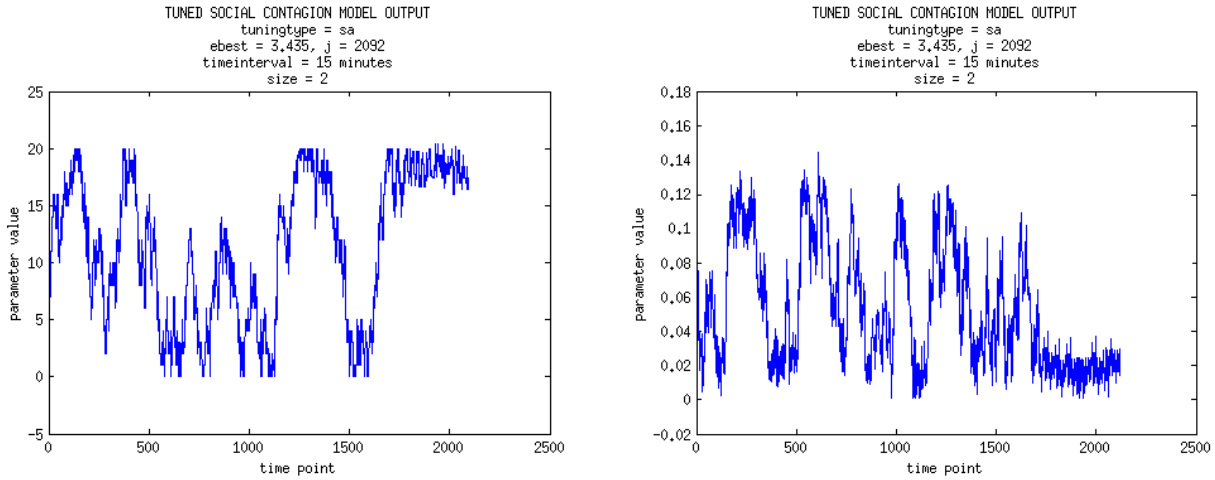


Figure 13: Respectively the parameters' alpha and beta values over time

For the Simulated Annealing we have choose the initial value for the parameter alpha equal to 10 and for beta of 0.7, since both are in the middle of the search space; in Figure 13 it can be noticed that in the last 400 iterations of the method the range in which the parameters are explored is from 15 to 20 for what concerns alpha and from 0.01 and 0.02 for what concerns beta. This refinement search is translated in a decrease in the energy as it can be seen from Figure 14;

the best energy value is higher at the beginning (32 vs. 21) but the one obtained in the end, is less than the one obtained with the Linear Search since the value that fits most the model against the data is 0.013 which was impossible to obtain with the previous search method due to the granularity we have chosen.

The value found for alpha is the same found with the Linear Search method, 17.

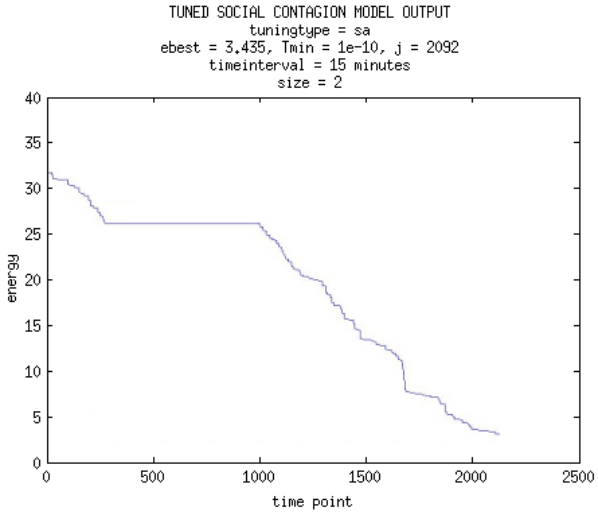


Figure 14: Best energy value evolution during the Simulated Annealing

From Figure 15 and Figure 16 we can see that with this value for beta the model follows better the real behavior of the agents' states up to noise errors in the data gained; for what concerns the opinion of the agents about the Jazz music, we can find also here that even with the new value found for beta the model doesn't validates well against the data, which is the main reason why the best energy level is quite high even in this case.

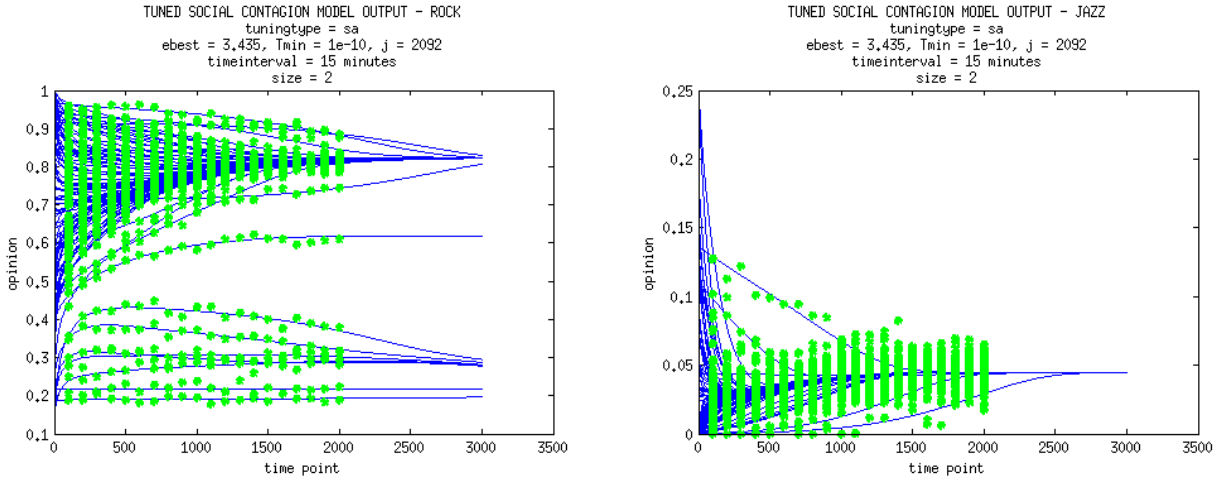


Figure 15: Output with the best parameter found with the Simulated Annealing method with the marker for the real agents' states respectively for Rock and Jazz opinions

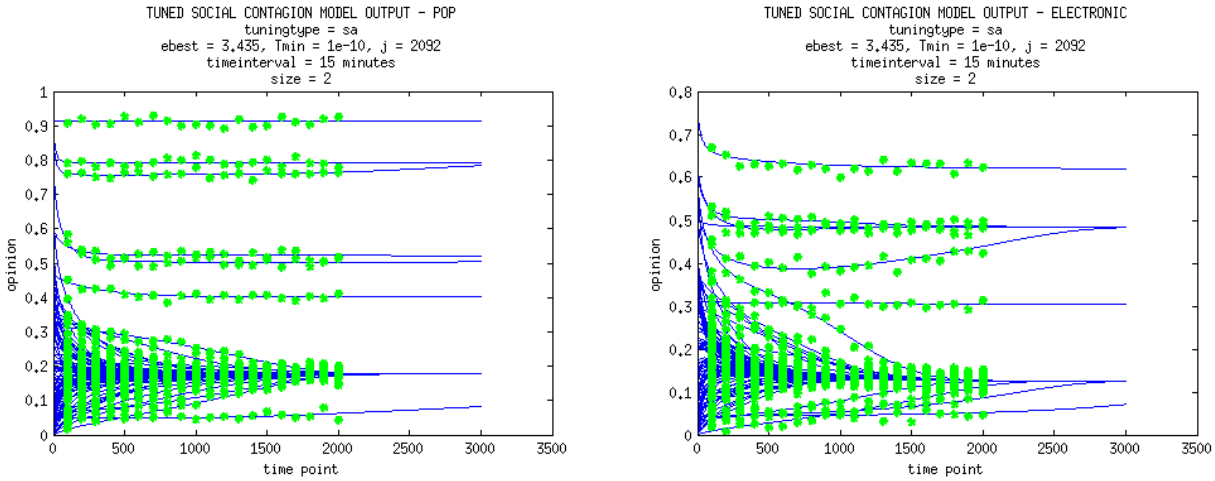


Figure 16: Output with the best parameter found with the Simulated Annealing method with the marker for the real agents' states respectively for Pop and Electronic opinions.

	Best value α	Best value β	Corresponding energy value
Linear Search	17	0.010	4.168
Simulated Annealing	17	0.013	3.435

Table 2: Comparison between the values obtained with Linear Search and Simulated Annealing method for the parameters α and β their corresponding energy values

Conclusions

We can conclude that the musical tastes affects the formation of clusters and contribute to modify the shape of the network in this sense; the model presented in [1] with our specific instantiation describes quite well the behavior of a Scale Free network of users at least with the amount of data that we collected.

We can also assume that if the users do not have a clear opinion about a certain kind of music (with opinion values which in our model are less than 0.25) they simply keep more or less their opinion; this behavior can be explained if we think that if in a network there are no agents, or really few which posts songs or artists of a certain kind, the impact is not strong enough to realize well the contagion process.

References:

[1] “Modeling and Analysis of Social Contagion Processes with Dynamic Networks” - J.Treur, A. Sharpanskykh

[2] “Different Types of Social Agent Models” - J.Treur

[3] “Social Networks: an Overview” - J.Treur

[4] Official reference for Last.fm API - http://www.lastfm.it/api

[5] Official reference for GEFX syntax - http://gexf.net/format/