Georgia Institute of Technology
CSE 6730

Simulation of rumor spread in social network

Project Report

Haomin Lin
(hlin374@gatech.edu)

Yuntian Zhang
(yzhang3469@gatech.edu)

Zichao Feng
(zfeng78@gatech.edu)

February 2020

1 Simulation description

1.1 Problem statement and SUI

1.1.1 Problem statement

Rumor spreading is an important factor in social communication, which has triggered vast research interest. It is much more complex than epidemic spreading or news spreading since there are uncertainty during the process. Sometimes a person doesn't believe the rumor at all, sometimes he buys it and becomes a rumor spreader. The probability will affect the result of the rumor spread. Currently, SIR models[1] are most popular for this sort of simulation. Also, there are some simulations focusing on large-scale social network, like SUPE-Net model[2]. However, there are several limits among current study :

(1) With social network being dynamic, it usually has a lot of interaction. The complexity of the model is much higher than a static model. And a parallel model is needed to simulate this properties[3].

(2) Social network usually involves higher participation, which makes the simulation harder to execute due to its large scale. Some large-scale models even need large scale parallel computers to simulate.

(3) In a social network system, the outcome of rumor spreading can be largely different if some uncertainties happens. Most of the models are built with a lot of assumptions, which makes the model less credible.

All these properties bring more difficulties to building an accurate social network simulation, which will also be the biggest challenges for our project. Through this simulation, we plan to study the rumor spreading in social networks of a moderate size. With fixed time or size, we want to see the power of rumor spreading with specific parameters set. Through this, we can also derive some strategies against rumor spreading from the results.

1.1.2 SUI

As mentioned, the system under investigation is a social network consisting of spreaders, truthers, and stiflers. Generally, there are two types of this system in our simulation, one is the open systems, which has no upper limit for the size of the network, constrained by the simulation time cap. And the other one is the restricted system, which has a set size.

In the SUI, the initial system has several spreaders and ignorants who are the spreaders' ”ignorant” friends, possibly with truthers who are introduced into the system at different moments. This means some people knows some rumor while some others know the truth against the rumor with or without delay. And the initial ignorants also bring ignorants to the network when they are contacted for the first time. These friends are within 3 unit distance to who introduces them into the system.

And every person only has limited ”friends” to communicate with. If they've participated in too many conversations, they won't make new conversations any more.

With the initial state of the rumor, the system begins to run with interactions. And the possibility one turns another type of person depends on the people around him. Generally, if there are more spreaders than truthers, the receiver in the conversation will incline to hold the same opinion as the talker in the conversation.

However, since some talkers may have overlapped friends, as the friend is talking with others, this talker will have to wait until the conversation is over to have a chance to start his conversation, as is normal in social networks. The two have been in a conversation will not talk again in the future.

When 2 people are talking, the length of the conversation is assumed to follow the pattern of a common phone call. And the outcome of the conversation is decided by the people surrounding them on the intimacy map. In most cases, the more Spreaders/Truthers/Stiflers around him, the more possible he will become one of the majority. Then, once the conversation finishes, if there's no other people waiting to talk to a spreader or a truther, we assume the spreader or truther will have a rest for 180 time units, then starts to seek another conversation.

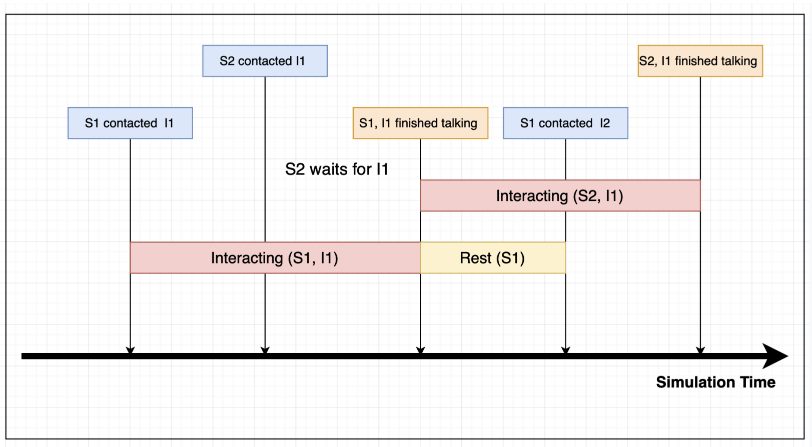


Figure 1: an snapshot of the entities' behavior

In both systems, the simulation will stop if all the people in the network becomes stiflers or ignorants. However, in the open system, the size of the system can grow continuously, and stops at the upper limit of the simulation time. These two systems can be used to study the rumor spreading in different kinds of social networks.

1.2 Related work

While most SIR models only involve the interaction between spreaders, ignorants and stiflers, our model includes one more role in the social network called truther to refute spreaders. Similarly, In the model mentioned in Laijun Zhao's paper about SIHR rumor spreading model in social networks[4], it introduces a new group called hibernators. The concept of introducing this new role is the same as our incentive of introducing the truther. While the difference is that the hibernators mentioned in the paper above won't actually influence the ratio of different S-I-R people in the social network but our truther will.

The SIR model was first introduced by Daley and Kendall[5−7] who proposed a standard rumor spreading model. The major shortcoming of the DK model is that it doesn't consider the effect of network typologies on the dynamic behaviors of rumor spreading. A lot of models proposed like Borge-Holthoefer model[8,9] and Zhang's model focusing on an online social blogging platform still have similar question. In paper related to SIR Rumor model in recent studies, several trust mechanisms are introduced. However, the model is usually hard to perfectly simulate the true one, some research uses alternative ways to make it happen. For example, Ya-Qi Wang et al.[10] mentioned a way that propose some rate to show the proportion of the trusted neighbors in Figure 2.

In our project, We assume that people are more possible to listen to their intimates(who are closest to them in the intimacy map).

In the paper *Theory of rumour spreading in complex social networks*[11], the author introduces a dynamic probability calculation from Markov chain mean-field equations, that is, the probability of a node (a person) gets con-

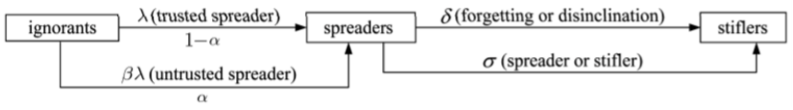


Figure 2: Structure of SIR rumor spreading process.

verted from an ignorant to a spreader is decided by the state of his n-nearest node. This concept is reasonable since it reflects the fact that every one tends to believe what their most close friends believe about a rumor or other news. The author also gives us an equation based on time interval t.

However, some other studies just simplify this question by giving each one in the network a equal number. The author in *Why Rumors Spread Fast in Social Networks*[12] believes that since the huge scale of the network, the average speed can be used to replace each individual nodes. This largely reduces the complexity of the model.

In [4], the author also mentioned this question by saying that the rate of if an ignorant buys the rumor largely depends on its background. He used the Chinese salt-buying frenzy case to strengthen his point of view.

1.3 Conceptual model

According to the SUI, we derive a conceptual model to include all the key factors in our simulation system:

Firstly, we introduce a class called ”User”. In our assumption, this ”User” class can be regarded as resources as well as customers. Since it ”uses” other users in the system as a resource to start a conversation and is ”used” as a resource for conversations in return.

Hence, as a resource, the system has the entity as ”receiver”. This entity has several attributes including ”freetime”, which indicates the time for next conversation to start; ”occupy”, indicating whether this receiver is in a conversation in current simulation time. And the ”receiver” is also a queue entity, with attribute ”wait_num” that indicates the current number of users waiting to talk to him, showing the nature of queue in this entity

While as a customer, the entity is called ”talker”, which has entities ”numFriend”, which restricting the energy of the customer: once it’s exceeded, one can’t start a conversation by himself anymore. Also a ”record” attribute to include all the partners one has, to eliminate repeated talks. After a talk, if there aren’t anyone waiting to talk to the talker, the talker will be placed to another resource, also a group entity ”rest period”, where the customer rest for 180 time units.

As a user, ”talker” and ”customer” combined has some attributes to define themselves: ”value” and ”Type” classifies them as Ignorant, Spreader, Truther, and Stifler believing the rumor or the truth; ”dists” that show their relationship in a intimacy distribution map; ”P” that contains the possibility that each type conversion may occur for them.

There are two kinds of activities in this system. One is ”rest”, triggered when a conversation finishes with the talker having no users waiting for him. This activity lasts for 180 time units. The other one is conversation. This activity can be triggered only when two users in the pair are both available, i.e, their ”wait_num” are both 0 and neither of them are occupied. And there’s an action named ”makeComb”, this action summons all the available Spreaders and Truthers to make conversations, altering the ”wait_num” and ”record” of both sides.

Concluded from entity and activities, there are several constants and parameters to mention: Rest_time equals

180, which is a constant that defines the length of the rest activity; Conversation_length, which follows the distribution from our collected data; Conversion possibility, which is a random number in (0,1), used to be compared with the possibility of different kinds of conversion and decide whether the conversion happens.

2 Simulation software

In the simulation software, we've made some specific modifications after the checkpoint:

Firstly, we modified the conversion possibility distribution of users participating in the conversation. Inspired by the Probabilistic Relational Classifier[13], we implement the following formula to calculate the possibility of each kind of role conversion according the possibility parameter in one's 10 closet users (represented as j in the formula):

$$P(\text{User}_i \text{ converted to Type}_t) = \frac{1}{10}\sum_{j}^{10} P(\text{User}_j \text{ converted to Type}_t) \tag{1}$$

By using this formula in the function "interact", we calculate the possibilities every time a conversation begins to make decisions. As for the initial members of the network, we assume the the active users will never turn to the opposite side in the beginning. While the ignorants has the possibility of conversion according to the assumed credibility of the rumor.

What's more, we also add a "rest" function comparing to the software at checkpoint to make active users (Spreaders and Truthers) to rest for some time after conversations before they start new ones, which makes the system work more realistically.

With all these updates, the software works as shown in this graph:

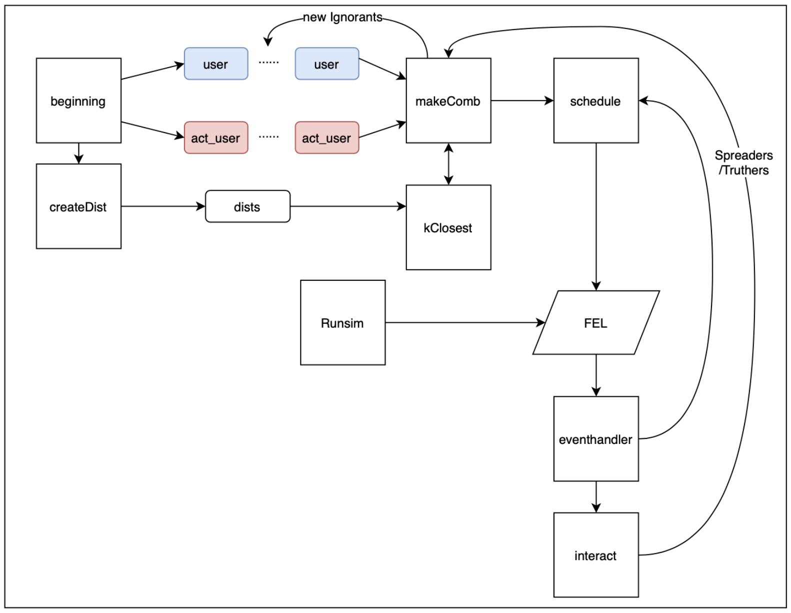


Figure 3: Software Diagram

Spreaders and Truthers with their ”ignorant” friends are introduced to the system with ”createDist” in the ”beginning” function. Then, the ”act_users” are summoned by ”makeComb” to make new conversations with the help of ”kClosest” and schedule them via ”schedule” in the FEL. As the ”Runsim” function called, the conversations are processed one by one. ”eventhandler” will reschedule the ”begin” events if anyone in the conversation is occupied and push the qualified conversations to ”interact” to carry out the conversation. Once an active user is available, they will be summoned by ”makeComb” to make new conversations. When no user in the systems is an active user, the simulation will stop. In another case, the simulation stops with simulation time is over the limit.

Also, in the simulation software, we can also generate visualizations with a machine that has a display, including observing the changing of some parameters in the simulation run and the snapshot of the system. What's more, it also calculates Confidence Interval with results collected from hundreds of iterations.

3 Data ultilization

In the project, we incorporate the data into the system as several parameters:

One is the length of the conversation. This data is derived from the statistical results of the length of conversations over the phone^{14}. We approximate the conversations in our system to the phone talks and make its length as this distribution:

$$\text{Length}\ (p) = \left\{ \begin{array}{lr} 15 * var & p < 0.116 \\ 15 + 22.5 * var, & 0.116 \leq p < 0.251 \\ 37.5 + 22.5 * var, & 0.251 \leq p < 0.419 \\ 60 + 60 * var, & 0.419 \leq p < 0.728 \\ 120 + 60 * var, & 0.728 \leq p < 0.835 \\ 180 + 60 * var, & 0.835 \leq p < 0.868 \\ 240 + 60 * var, & 0.868 \leq p < 0.945 \\ 300 + 300 * var, & 0.945 \leq p < 0.986 \\ 600 + 600 * var, & 0.986 \leq p < 0.994 \\ 1200 + 2400 * var, & \text{otherwise} \end{array} \right\} \tag{2}$$

where var is another random variable. And this distribution is implemented in the simulation software as function ”interval”.

Another one is the number of friends distribution from an article on Gallup[15], which gives the statistical results of the number of friends one could have, which is shown as this distribution:

$$\text{numFriends}\ (p) = \left\{ \begin{array}{lr} 0 & p < 0.002 \\ 1, & 0.02 \leq s < 0.08 \\ 2, & 0.08 \leq s < 0.16 \\ 3, & 0.16 \leq s < 0.27 \\ 4, & 0.27 \leq s < 0.38 \\ 5, & 0.38 \leq s < 0.55 \\ 6 + 4 * var, & 0.55 \leq s < 0.73 \\ 10 + 5 * var, & \text{otherwise} \end{array} \right\} \tag{3}$$

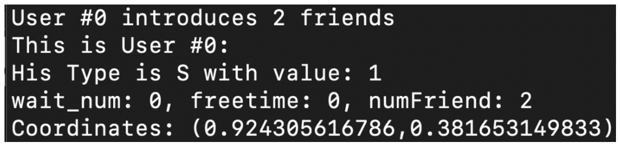
where var is another random variable. And this distribution is implemented in the simulation software as function ”numFriends”.

4 Verification and Validation

4.1 Verification

With the data implemented in the software, we firstly verify our simulation system.

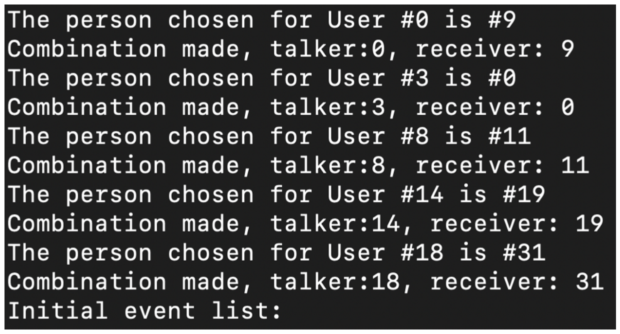
We firstly generate several spreaders and truthers with their ”ignorant” friends labeled as “active users” in the network.



```
User #0 introduces 2 friends
This is User #0:
His Type is S with value: 1
wait_num: 0, freetime: 0, numFriend: 2
Coordinates: (0.924305616786,0.381653149833)
```

Figure 4: User generation

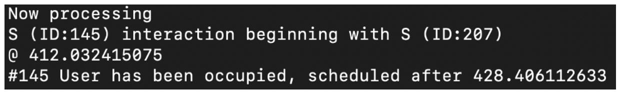
After initialization, the simulation begins to run by making combinations between users and schedule the interaction in the Future Event List (FEL), then the simulation begins.



```
The person chosen for User #0 is #9
Combination made, talker:0, receiver: 9
The person chosen for User #3 is #0
Combination made, talker:3, receiver: 0
The person chosen for User #8 is #11
Combination made, talker:8, receiver: 11
The person chosen for User #14 is #19
Combination made, talker:14, receiver: 19
The person chosen for User #18 is #31
Combination made, talker:18, receiver: 31
Initial event list:
```

Figure 5: User combination for interaction

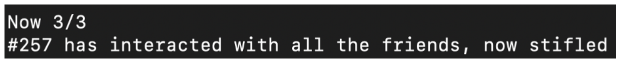
In the scheduling process, once an active user encounters a user who he wants to talk with is already occupied, he will have to wait until the current conversation ends, queuing after someone who is currently in the talk.



```
Now processing
S (ID:145) interaction beginning with S (ID:207)
@ 412.032415075
#145 User has been occupied, scheduled after 428.406112633
```

Figure 6: Queuing action

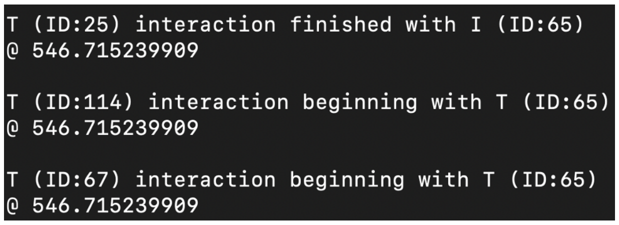
In the combination making process, we implement a 2D intimacy map, storing the coordinates in a list “dists”, corresponding to the users. When making combinations, we calculate several coordinates closest to the current talker, and choose one randomly to make the combination.



```
Now 3/3
#257 has interacted with all the friends, now stifled
```

Figure 7: Stifled user

When handling an event, if both sides in the conversation are not in a talk at the moment, the simulation will schedule a “finished” event for them and decide the length of conversation according to function “interval”. However, a “begin” event will be scheduled with one of the users in a conversation, indicating that this pair has to wait until the currently talking person finishes. The waiting and occupying state are reflected in the “wait_num” and “occupy” attributes in the “User” structure. And the new waiting ones will have to wait after those who come earlier.



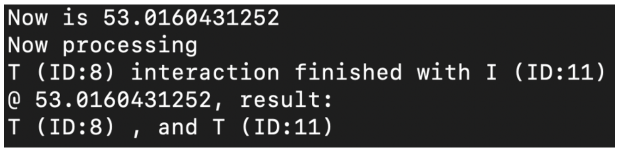
```
T (ID:25) interaction finished with I (ID:65)
@ 546.715239909

T (ID:114) interaction beginning with T (ID:65)
@ 546.715239909

T (ID:67) interaction beginning with T (ID:65)
@ 546.715239909
```

Figure 8: Multiple queueing entities

In the “interaction”, we use the formula introduced in Data utilization section to decide the conversion result. And the result will reflect in the user’s “Type” attribute. Type “I” can be converted to “RS”/“RT”/“S”/“T”, and “T” can be converted to “RT”, “S” can be converted to “RS”. “I”, “RS” and “RT” will not trigger new conversations.



```
Now is 53.0160431252
Now processing
T (ID:8) interaction finished with I (ID:11)
@ 53.0160431252, result:
T (ID:8) , and T (ID:11)
```

Figure 9: Interaction result

4.2 Validation

Then, we run experiments on our simulation software to validate our simulation.

Since the model we try to build is similar to the model built in the SIHR rumor spreading model by Zhao Laijun et al., we compare the behavior of our model with the SIHR model:

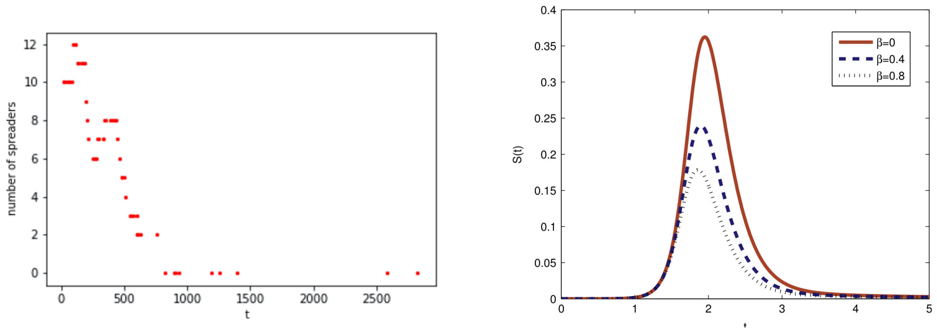


Figure 10: Comparison on the variation about spreaders, left: our model, right: SIHR model

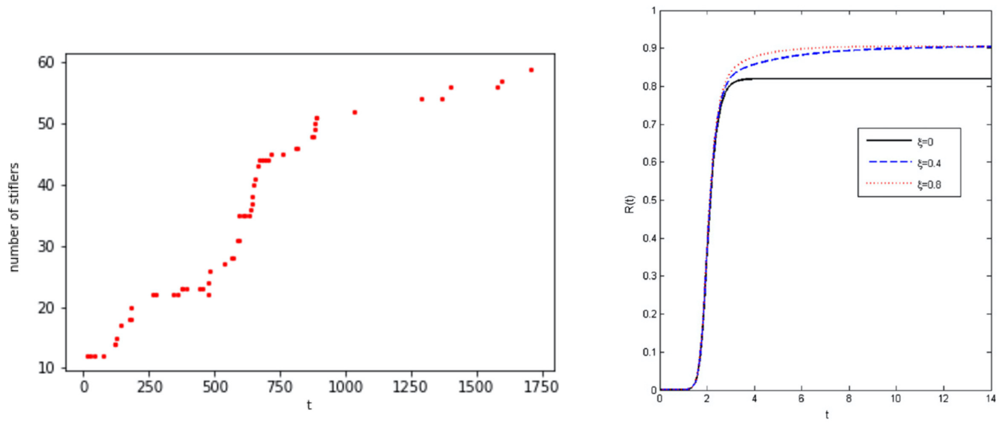


Figure 11: Comparison on the variation about stiflers, left: our model, right: SIHR model

We can see that, even though the initial setting is different, with the increasing of the simulation time, the number of spreaders and truthers will both decrease after reaching a peak. While the group of stiflers keeps growing.

And this result is also similar to the real case. Usually, when the spreaders or truthers feel unnecessary to keep spreading the rumor or refute it, the rumor spreading behavior will naturally stop.

Also, through behavior validation, we can also see that, in this simulation, the rumor spreading can be undermined more greatly as the S/T rates drops, which fits the real case.

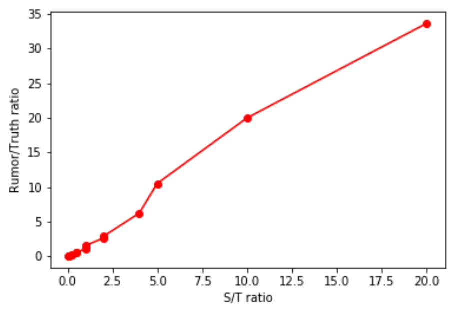


Figure 12: Rumor spreading undermined phenomena

5 Output analysis

5.1 Steady state studies

To study the steady state. We found that the percentage of ”touched” users in the social network presents the pattern of gradually becoming steady.

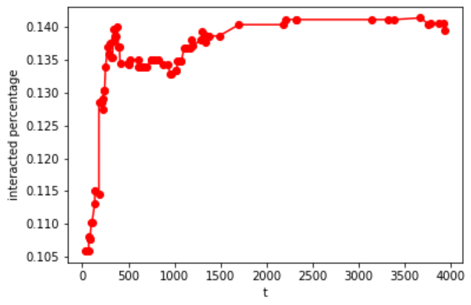


Figure 13: Steady state

As shown in the graph, in most cases, this ”touched” percentage always climbs up in the beginning, then stops growing greatly at a relatively steady value.

In this period in the simulation, the reason for the low percentage of interaction can be that the system is just built and the users in the network haven't been sufficiently mixed. Once the interaction has happened enough times, according to the possibility distribution, the interacted persons will take up a approximately fixed portion of the total population.

What's more, for steady state studies, there are two cases we need to consider for the time of completion for the simulation. If it's a restricted system, simulation will stop when all the users in the system are inactive. While when it's an open system, simulation stops when the FEL holds no event that has timestamp less than the set simulation time. For those two cases, the time when the simulation ends can be determined via the timestamp posted by last processed event.

5.2 Confidence interval

To study the power of rumor spreading and the outcome of adding truthers to contradict rumor, we compute these two parameters, and derive their confidence interval:

(1) Interacted rates
From steady state study, we can know that interacted rate is the data that will come to a steady state in the simulation. To test the power of rumor spreading, we run several experiments to see the best value for accepting a point estimate, and choose the number of 200 iterations as the best value, for the improvement begins to be trivial after this.

Number of runs	Point Estimate	Standard Deviation	ζ	CI$_{min}$	CI$_{max}$	ζ/mean
20	0.139	0.002	0.004	0.136	0.143	0.032
50	0.142	0.001	0.002	0.139	0.143	0.016
100	0.142	0.001	0.002	0.140	0.144	0.014
200	0.142	0.001	0.002	0.141	0.144	0.011
500	0.142	0.001	0.001	0.141	0.143	0.007

(2) Rumor/Truth ratio

For Rumor/Truth ratio, which we found has a relatively unsteady performance, we also conducted a research on its confidence interval, and the result is shown here:

Number of runs	Point Estimate	Standard Deviation	ζ	CI$_{min}$	CI$_{max}$	ζ/mean
20	0.655	0.090	0.187	0.468	0.842	0.286
50	0.635	0.071	0.142	0.492	0.777	0.224
100	0.608	0.039	0.077	0.531	0.685	0.126
200	0.596	0.026	0.050	0.546	0.647	0.09
500	0.617	0.018	0.035	0.582	0.652	0.06

After comparison, we can find that it has best performance when running 500 iterations, which is within the computation power of our machine, so we use this for later experiments

5.3 Result analysis

With the above output analysis, we carry out several experiments to study the behavior in our system, which helps us derive some thoughts about the real system. According to the conclusion drawn from 5.2, we run every experiment 200 times.

Firstly, we study the percentage of the interacted people in our network with different allocation of initial spreaders and truthers. We run the experiments in a system with 1000 people, 10 spreaders and 20 truthers in the beginning of the simulation. We tried 3 different initial allocations, shown as: In the first one, the active users

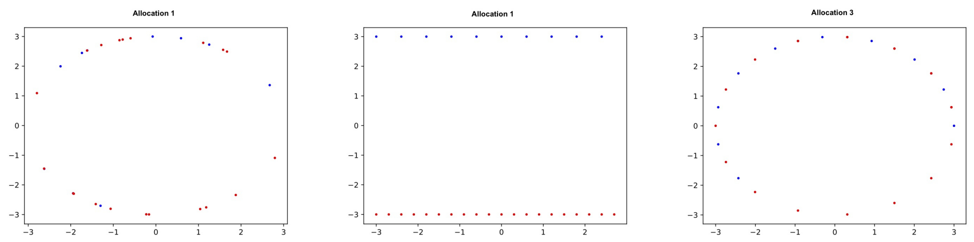


Figure 14: 3 different allocations for experiments

takes a place on a circle with diameter = 6. Users in the second allocation are placed separately in the two opposite areas. And users in the third allocation are placed on the same circle as that in the first allocation homogeneously.

Then we calculate the interacted percentage and Rumor/Truth ratio. The results are:

Allocation Type	1	2	3
Interacted percentage	0.139	0.139	0.143
Rumor/Truth ratio	0.567	0.773	0.561

We found that the implementation different allocations barely affect the percentage of people who have been interacted in the system. But separating the truther and spreaders will strongly undermine the power of the truthers who contradict the rumor.

After this, we turn to he outcome of rumor spreading with same simulation time in the same network while with different credibility of the rumor. Same, in the same system from last topic, with default possibility of conversion (persuasion) varying, we set the following experiments and get the result:

Credibility of the rumor	0.2	0.5	0.8
Interacted percentage	0.141	0.139	0.140
Rumor/Truth ratio	0.315	0.567	0.970

We found that the credibility won't affect the percentage of interacted people, but will undermine or boost the spreading of rumor. Once the rumor becomes more credible, the power of truthers will be undermined and vice versa. The credibility has great impact on the simulation.

The third topic of interest is the power of truthers to stop rumor spreading. We make the truthers introduced to the same system as that in the past to see the outcome. Since the simulation time for a restricted system won't last too long, we set the truther entering time (delay) as 0, 100, 500, 800, 1000 with the result:

Delay	0	100	500	800	1000
Interacted percentage	0.139	0.140	0.140	0.139	0.139
Rumor/Truth ratio	0.567	0.559	0.0.556	0.555	0.566

We found that the delay of the truthers' arrival won't make too much impact on the power of truthers, even if they arrive later in the system, since the spreader has limited ability of spreading rumor in our system, the spreading can be undermined by truthers as long as they can enter the system.

At last, we look into the outcome of rumor spreading with different simulation time in the same network. In this process, we set the system as an open system. Keeping the other parameters fixed, we run the iterations for several simulation time, getting results as:

Simulation time	1000	5000	10000	20000	50000
Final population	653.92	724.47	712.51	717.825	706.53

With ζ/mean 0.03, we found that the network size of an open system also has a bottleneck. Since the simulation itself has self-constrained nature, it's possible that the network stops growing at a specific point. And sometimes the simulation will end before the time limit. The major reason behind this is that the spreading power is limited, which cannot support the rumor to spread forever.

What's more, we also introduce a function to visualize the original state and the result of the users in the system, using blue dots to represent spreaders and green dots to represent stiflers who believe the rumor, while red and yellow dots are truthers and stiflers who buy the truth. An example is shown as:

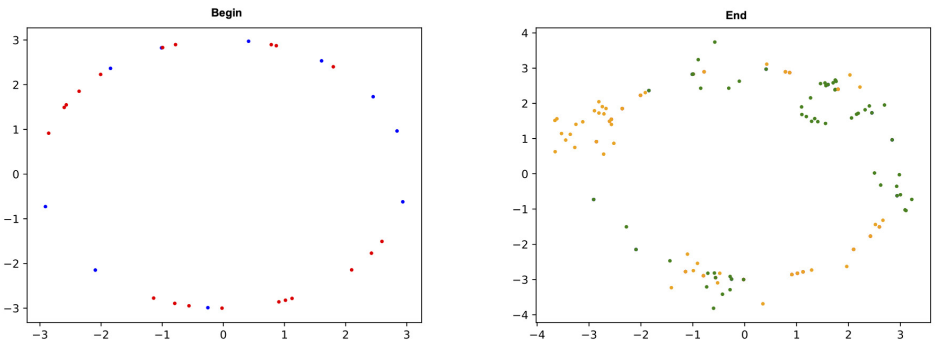


Figure 15: Initial and Final state visualization

6 Conclusion and limitations

In conclusion, we build a simulation system to simulate the rumor spreading in the social network. We introduce 4 different roles in this system, simulating the process where rumor spreads and gets contradicted in a social network. After trying different methods to calculate the conversion probability, we decide the best one inspired by Probabilistic Relational Classifier, which is one of the key parts of our system. We also utilized data from different sources to build up the behaviors of entities in our system.

With this system, we analyze the output as conditions vary. We found that the allocation of initial active users may affect the truther-spreader collision, separated distribution of two sides can make rumor easier to spread. Also, we find that the credibility matters in the spreading of the rumor, which largely slows or speeds up the rumor spreading. And we also find that the rumor spreading has self-restricted property. As the active users become stiflers, the rumor spreading will stop naturally instead of spreading continuously. This is also supported by Zhao Laijun et al.'s SIHR model. Through the simulation, we can be sure that a key factor to stop rumor spreading is to improve the ability of the people in the network to distinguish rumor, i.e, reduce the credibility of the rumor.

Inevitably, our simulation model holds some limitations. The most apparent one is the scale of the network. Due to the limitation of computation power, it's difficult for us to compute a network with large number of people (over 100 in the initial stage) several times to get ideal confidence interval and point estimate results. In the future, the data structure can be modified to make the program run faster. For example, Ladder queue[16] can be implemented. Also, a new way of selecting the conversation partner can also be used to make the system more realistic. The assumption of the intimacy map may actually constrain the power of spreaders to spread the rumor.

In this project, each member dedicated equally in building the simulation model as well as experiment design and execution, every idea in the project matters for reaching the final result.

References

[1] Laijun Zhao, Hongxin Cui, Xiaoyan Qiu, Xiaoli Wang, Jiajia Wang. "SIR rumor spreading model in the new media age," *Physica A: Statistical Mechanics and its Applications*, 1998, 392(4): 995-1003

[2] B. Hou, Y. Yao, B. Wang and D. Liao. "SUPE-Net: An Efficient Parallel Simulation Environment for Large-Scale Networked Social Dynamics," *2010 IEEE/ACM Int'l Conference on Green Computing and Communications* & *Int'l Conference on Cyber, Physical and Social Computing*, Hangzhou, 2010, pp. 628-635.

[3] Hou B, Yao Y, Wang B, et al. "Modeling and simulation of large-scale social networks using parallel discrete event simulation," *Simulation: Transactions of the Society for Modeling and Simulation International*, 2013, 89(10): 1173-1183.

[4] Zhao Laijun, Wang Jiajia, Chen Yucheng, Wang Qin, Cheng Jingjing and Cui Hongxin "SIHR rumor spreading model in social networks" *Physica A: Statistical Mechanics and its Applications*, 2012, 391(7):2444-2453

[5] Daley, D., Kedall, D, "Epidemics and Rumours" *Nature*, 1964, 1118(204)

[6] Newman, M. E. J. and Forrest, Stephanie and Balthrop, Justin, "Email networks and the spread of computer viruses" *Phys. Rev. E*, 2002, 66(3):035101

[7] Wang, Fang and Moreno, Yamir and Sun, Yaoru, "Structure of peer-to-peer social networks" *Phys. Rev. E*, 2006, 73(3):036123

[8] J. Gu, W. Li, X. Cai, "The Effect of the Forget-remember Mechanism on Spreading" *Eur. Phys. J. B*, 2008, 62(2):247-255

[9] De Lellis, Pietro, Jinxian Li, Yanping Hu, Zhen Jin, "Rumor Spreading of an SIHR Model in Heterogeneous Networks Based on Probability Generating Function" *Complexity*, 2019:1-15

[10] Ya-Qi Wang and Xiao-Yuan Yang and Yi-Liang Han and Xu-An Wang, ”Rumor Spreading Model with Trust Mechanism in Complex Social Networks” *Communications in Theoretical Physics*, 2013, 59(4):510-516

[11] M. Nekovee, Y. Moreno, G. Bianconi, M. Marsili, ”Theory of rumour spreading in complex social networks,” *Physica A: Statistical Mechanics and its Applications*, 2007 ,374(1):457-470.

[12] Doerr, Benjamin, Fouz, Mahmoud, Friedrich, Tobias, ”Why Rumors Spread So Quickly in Social Networks,” *Communications of The ACM*, 2012, 55:70-75. 10.1145/2184319.2184338.

[13] Zhen Han and Alyson Wilson ”Dynamic Stacked Generalization for Node Classification on Networks,” *ArXiv*, 2016, abs/1610.04804.

[14] Peilin Luo, Leopoldina Fortunati, Shanhua Yang, ”New technologies in global societies,” *World Scientific*, 2006:168.

[15] Joseph Carroll, “Americans Satisfied With Number of Friends, Closeness of Friendships” *Gallup News Service*, 2004.

[16] Tang W T, Goh R S M, Thng I L J, ”Ladder queue: An O (1) priority queue structure for large-scale discrete event simulation,” *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 2005, 15(3):175-204.