Design of Multi-Agent Systems - B20
Gossip Simulations with Semi-Random Strategies

Arjan Jawahier (s2762161)
Roeland Lindhout (s2954524)
Xabi Krant (s2955156)
Niek Naber (s2515970)

Final version, November 3, 2019

Abstract

Communication problems are a widely studied phenomenon. The gossip problem is a widely studied form of the communication problem. This study tries to find new, computationally inexpensive semi-random protocols to improve completion time of the gossip problem. To closely mimic real world problems, this study focuses on a version of the gossip problem which is distributed, parallel and uses two-way communication. Agents act on their own, without any form of centralized supervision. Multiple agents can act simultaneously in one time-step, and communication between agents is always two-way. The results of this study can be used to improve e.g. communication and agreement protocols in Blockchain. Our findings show that it is possible to improve on the minimum number of time-steps required to solve the problem by using newly developed semi-random strategies. Although the semi-random strategies result in a significantly lower completion time, the improvement is minimal. For solving this problem the random strategy is not optimal, but very viable, and easy to implement.

1 Introduction

In this paper we consider communication problems between agents. Several versions of this problem can be found in the literature, such as the marble problem [Lan54], or the gossip problem [HMS72]. Both problems are in essence the same, but in this paper we will focus on the latter, because of its simplicity.

1.1 Problem

The gossip problem is originally described as the following [HMS72; BS72]:

> *"There are n ladies, and each of them knows some item of scandal which is not known to any of the others. They communicate by telephone, and whenever two ladies make a call, they pass on to each other, as much scandal as they know at that time. How many calls are needed before all ladies know all the scandal?"*

We know that the minimum time needed when only 1 call can be made per time step, is $t(n) = 2n - 4$ [HMS72]. We will however consider a situation where multiple calls can be

made per time step. Each agent can only make one connection per time step. Communication is two-way, meaning that if an agent a tells its secrets to agent b, agent b will also tell its secrets to agent a. For every pair of agents a and b the connection to each other is the only connection they make in a single time step.

1.2 State of the art

According to [Lan54], the minimal completion time needed by a group of agents to complete the parallel, one-way gossip problem is τ where τ is the integer satisfying equation 1.

$$log_{r+1}n \leq \tau < 1 + log_{r+1}n \tag{1}$$

In this formula n is the number of agents, and r stands for the number of concurrent outgoing connections an agent can make per time step. This formula considers one-way parallel communication per time-step. For $n > 4$, Landau could however not mention a set of rules so that the completion time of the task reduces to this minimal completion time.
It is important to notice that this minimal completion time can only be reached in the centralized version of the gossip problem, "where the protocols tell the agents whom they have to call" [Coo+19]. The opposite of this is the distributed version, where "individual agents, on the basis of their own information, decide which other agent to call" [Coo+19].

1.3 New idea

We will take a look at the parallel version of the gossip problem, with r (the number of outgoing messages per time step per agent) being one, and two-way communication. Two-way communication can be thought of in this case as one-way communication with the simple extra restriction that *if any agent a calls agent b to tell its secrets, agent b must also call agent a in that time-step to tell his secrets.* Note that variable r only has to do with outgoing messages per agent. r is therefore still 1.
The new ideas proposed in this project are the introduction of semi-random strategies. [Lan54] mentions in his paper that the minimum completion time is attainable by using the optimal strategy. This implies that whenever the optimal strategy is not used, there is a probability that the minimum completion time will not be reached. The optimal strategy Landau speaks of however, is the strategy where each agent knows which agent to speak to in order to reach that optimal time, and thus the centralized version of this problem. In real life, this sort of knowledge is usually unattainable. We will therefore investigate the distributed version of the gossip problem. The fact that r is chosen to be 1, and communication is chosen to be two-way, resembles real life the best. When we consider agents to be humans which are all in their own isolated chambers, or simple processor threads for example, only one connection can be made per time step, and information can easily be exchanged between two agents, making the type of communication two-way. The two way connection can resemble a telephone line in the case of humans, or a web socket connection in the case of processor threads (This is a two-way connection because of the full-duplex characteristic [Net]). Distributed strategies that decrease completion time of the gossip problem, can be used in communication and agreement protocols in systems like Blockchain, to be more time efficient.
The methods proposed in this paper have to do with the introduction of semi-randomness in the strategies the agents make use of. In these strategies, agents use randomness as well

as a set of rules when they consider whom to call. The effect of these new strategies, and some existing strategies, will be studied by measuring the time needed for the completion of the task, and comparing these times with the theoretical minimal completion time for one-way communication (equation 1). We can compare optimal completion times of two-way and one-way communication problems because they are very similar, as explained before in this section. In our two-way version, there is still only one outgoing connection per agent possible. This makes it in essence possible to create the same communication graphs with one-way communication, as we create with two-way communications.

2 Method

2.1 Simulation model

The gossiping task can be represented by a graph containing nodes and edges. The nodes represent the agents and the edges represent the possible lines of communication between the agents. Landau talks about different kinds of graphs, where the number of edges is not equal for each node [Lan54]. In this paper, only a fully connected graph structure is considered (see Figure 1). Each agent is then able to communicate with every other agent. This change was made because it seemed to model real world applications better (where every server/thread/human is able to call every other agent. Here we assume that every agent has every other agents addres/phone number).

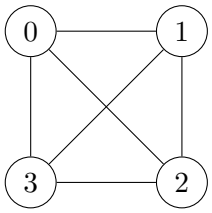

Figure 1: An example of a fully connected network of agents ($n = 4$). Each node represents an agent in the network. Each edge signifies a possible connection between agents, through which they can communicate their secrets to each other.

In Landau's work, computer simulations were performed using only 3 to 6 agents [Lan54]. The reason for this is that the computers of that time were not nearly as fast as the computers in our time (2019). This project will also handle an increased numbers of agents.

Simulations will be run using $n = 10$, $n = 50$, $n = 100$ and $n = 500$ agents. Different communication strategies will be implemented in each agent for different simulations. There exist a few general rules which remain the same in all simulations:

- Agents can only make $r = 1$ connection per time step with another agent.
- Agents always use two-way communication
- If an agent is already called by another agent, it loses the ability to call another agent during that time-step itself, because it already has a connection.
- The current time-step is known by all agents.

- To decide which agent can make the first call in a time-step, the order of a list of agents is randomized. The first agent in this randomized list makes a connection with the agent of his choice. Then the two agents that now have a connection cannot call again in this time-step. Now the next agent in the randomized list can make a connection. This process repeats itself until all agents (or in the case of an odd number of agents: all agents but one) have a connection with another agent. It should be noted that some strategies might restrain agents from calling other agents. In a case such as this, not all agents will have a connection with another agent during a time-step.
- All agents are initialized with an id. An agent can be called by this id. Id's go from 0 to n-1 where n is the total number of agents. The id's are common knowledge among all agents and can be used in their strategies.
- If agent a calls agent b while agent b already has a connection with some other agent, agent a can continue trying to call other agents during that time step. This simulates a bouncing connection request. We consider the time lost by making a bouncing connection request neglectable, because it can be done much faster than the actual information transfer. If an agent makes a connection request that bounces, it can therefore in the same time-step call another agent.

2.1.1 Strategies

Multiple strategies will be used in the experiments. We use already existing strategies to compare to our devised strategies. Existing strategies that are used can be found in Table 1. Table 2 shows all new strategies that were implemented.

	Explanation
Name	Agent a calls...
Random	a completely random agent b
Call me once	a random agent b iff he has not called b before
Learn new secrets	a random agent b of which he does not know the secret yet
Token	a random agent b iff agent a has a token. The token is given to agent b.
Spider	a random agent b iff agent a has a token. The token of agent b is taken by agent a.

Table 1: All existing strategies with their explanation. The strategies that we used are an interpretation of the strategies described in [Dit+19; AGH15; DFS14].

Name	Explanation
Multiply	Agent a calls an agent b with id corresponding to formula $callId = ownId*(t+1)\ \%\ n$. If however agent b already has another connection, agent a will increment the calculated id of the agent he wants to call by 1 until he finds an available agent.
Bubble	The agents call the agent with their id plus $2^{timestep}$. This results in bubbles of $2^{timestep}$ agents. Each agent know all secrets of the agents in their bubble. This bubble is expanded in each time step. Agents that do not fit into a bubble call a random available agent.
Min Secrets	Agent a calls agent b with the lowest amount of secrets, that has not been called yet
Max Secrets	Agent a calls agent b with the highest amount of secrets, that has not been called yet
Balanced Secrets	Agent a calls agent b if agent b has the most secrets and has not been called in the past 5 time-steps. If agent a knows all secrets, agent a calls agent c with the fewest secrets

Table 2: All newly implemented strategies with their explanation.

The x *Secrets* strategies, described in the table above, use s_i = the number of secrets that an other agents i knows. This number is an approximation based on what agents pass along. When an agent communicates with another agent, they share the number of secrets they both know. This is stored in their memory. When the agent then communicates with another new agent, they can also pass along all known or approximated numbers s_i collected through the full history of time-steps. If two agents share their knowledge of s_i, and have conflicting numbers, the highest of these two numbers s_i is decided upon, as the number of known secrets for an agent can not decrease over time. The maximum s_i is therefore always a better approximation than the minimum s_i.

2.2 Implementation details

The simulations are done using a program written in Python 3.7. The libraries used are: NetworkX 2.4, to generate graphs easily, Dash 1.5.1 to view and update the graph with ease, matplotlib 1.5.1 for the easy construction of histogram images and pandas 0.25.2 to store the results of the experiments in an easy to use DataFrame format.

2.3 Experiment design

For every strategy, 1000 simulations are run for each number of agents n where $n = 10$, $n = 50$, $n = 100$ and $n = 500$. In each set of simulations, all agents share the same strategy. Completion time is measured in time-steps taken.

To complete the gossiping task the fastest, agents should talk to many different agents. If each agent were to keep connecting to the same set of agents, no new secrets would be communicated. The *Random* strategy is already expected to be performing quite well, because it divides calls from one agent to multiple different agents. This makes communication diverse.

It is also expected to be hard to improve upon this random strategy. Essentially, the *Random* strategy is the least restrained strategies in the list of strategies, given in Tables 1 and 2.

3 Results

3.1 Experiment findings

The results of all strategies that we used can be found in Table 3. The comparison between the strategies and the optimal number of time-steps, calculated from equation 1 [Lan54], can be found in Table 4. Figure 2 shows all results from Table 3 together with τ_{opt} for different sizes of n. Figure 3 shows the results of the best 6 strategies in Table 3 together with τ_{opt} for different sizes of n.

Strategy	mean ± sd τ n = 10	mean ± sd τ n = 50	mean ± sd τ n = 100	mean ± sd τ n = 500
Random	5.47 ± 0.82	9.13 ± 0.48	10.37 ± 0.50	13.09 ± 0.29
Call Me Once	5.50 ± 0.82	9.13 ± 0.46	10.37 ± 0.49	13.11 ± 0.31
Learn New Secrets	**4.82 ± 0.56**	9.36 ± 0.63	10.93 ± 0.66	14.09 ± 0.49
Token	18.76 ± 9.71	95.35 ± 54.98	183.01 ± 112.76	NA
Spider	18.96 ± 10.07	100.19 ± 59.45	182.54 ± 112.27	NA
Multiply	5.56 ± 0.91	9.25 ± 0.57	10.59 ± 0.69	13.36 ± 0.58
Bubble	5.30 ± 0.74	8.91 ± 0.53	10.10 ± 0.38	**12.81 ± 0.41**
Min Secrets	4.89 ± 0.57	**8.58 ± 0.51**	**10.07 ± 0.27**	13.03 ± 0.17
Max Secrets	45.05 ± 30.37	902.46 ± 375.85	NA	NA
Balanced Secrets	5.59 ± 0.84	12.02 ± 0.87	14.94 ± 0.78	21.56 ± 0.66

Table 3: Table that displays the mean completion times of all strategies, shown for $n = 10$, $n = 50$, $n = 100$ and $n = 500$ agents. NA(not available)-fields were too computationally expensive to run for the given number of agents.

Strategy	τ/τ_{opt} n = 10	τ/τ_{opt} n = 50	τ/τ_{opt} n = 100	τ/τ_{opt} n = 500
Random	136.75%	152.17%	148.14%	145.44%
Call Me Once	137.50%	152.17%	148.14%	145.67%
Learn New Secrets	**120.50**%	156.00%	156.14%	156.55%
Token	469.04%	1589.17%	2614.43%	NA
Spider	474.02%	1669.83%	2607.71%	NA
Multiply	131.50%	153.83%	149.71%	148.44%
Bubble	134.50%	148.50%	144.29%	**142.33%**
Min Secrets	122.25%	**143.03%**	**143.86%**	144.78%
Max Secrets	150.41%	10518.18%	NA	NA
Balanced Secrets	139.75%	200.33%	207.01%	239.56%

Table 4: Table that displays the ratio of the average completion time (τ) over the optimal completion time (τ_{opt}), shown for $n = 10$, $n = 50$, $n = 100$ and $n = 500$ agents. $\tau_{opt} = 4$, 6, 7 and 9 for respectively $n = 10$, $n = 50$, $n = 100$, $n = 500$. NA fields do not have a τ/τ_{opt} ratio, because they were too computationally expensive to run.

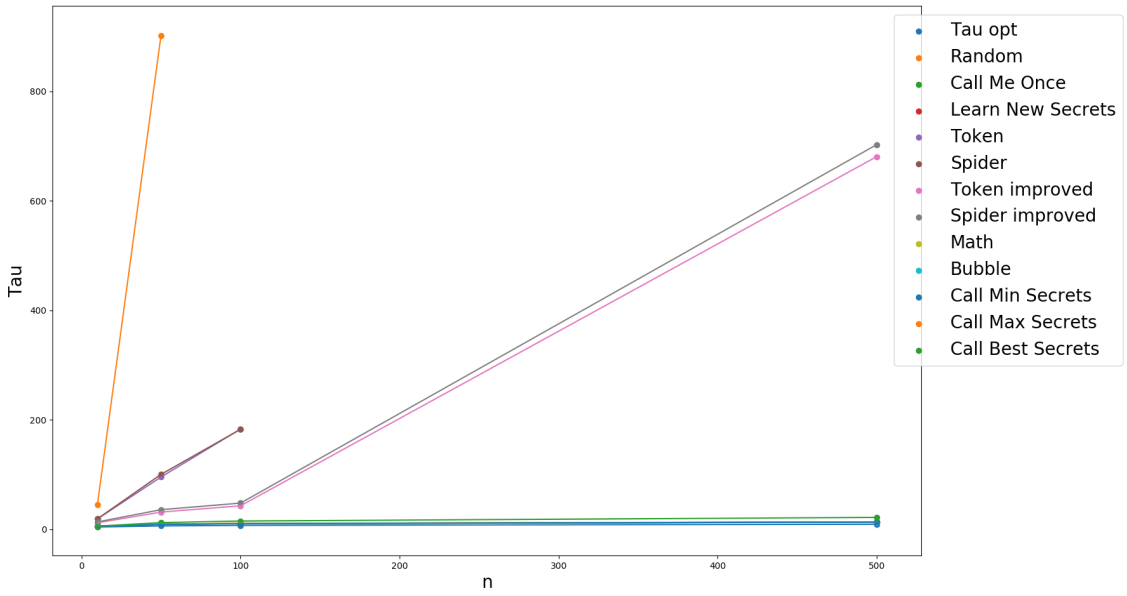

Figure 2: Average completion time for strategies, for different numbers of n. For every point, 1000 simulations were run. The *Tau opt* line in the graph is the theoretical optimal completion time. Note that the points in the graph are the actual mean measures of Tau. The dots of a strategy are only connected to improve readability.

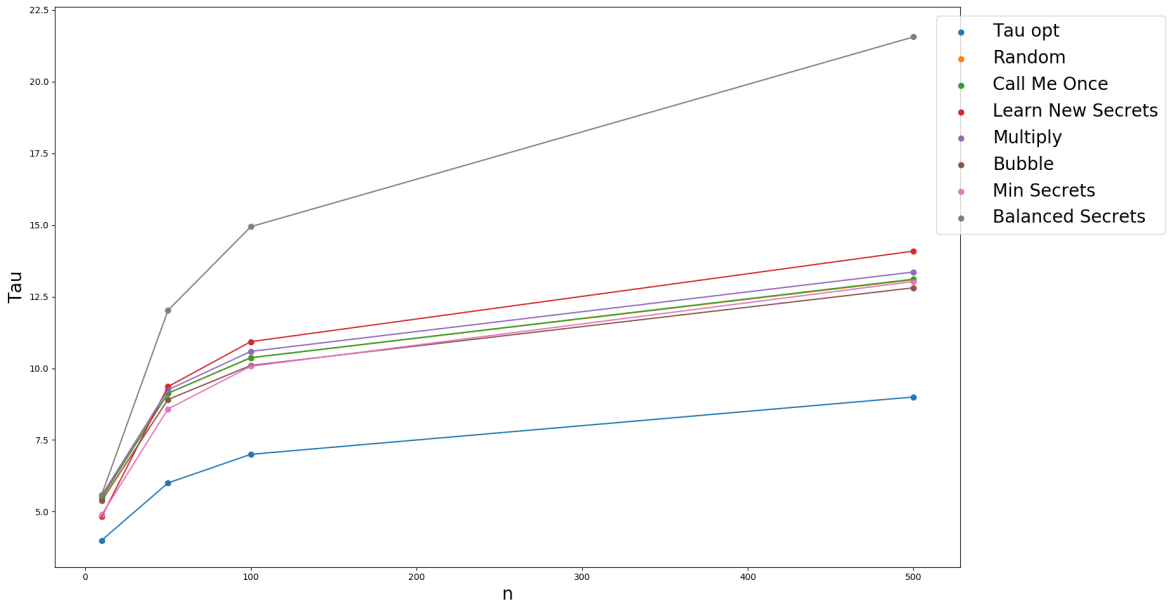

Figure 3: Average completion time for top 6 performing strategies, for different numbers of n. The *Tau opt* line in the graph is the theoretical optimal completion time. Note that the points in the graph are the actual mean measures of Tau. The dots of a strategy are only connected to improve readability.

3.2 Interpretation of findings

If we consider all of the simulated strategies, the ratio between τ and τ_{opt} varies from 142 to over 10000 percent for $n = 50$. There are 6 strategies that result in a $\frac{\tau}{\tau_{opt}}$ ratio below 150% for $n = 500$: *Random, Call Me Once, Learn New Secrets, Math, Bubble,* and *Call Min Secrets* (these strategies can be found in the graph in Figure 3). Compared to the *Random* strategy, there was a significant 3% gain possible in the τ/τ_{opt} ratio ($p < 0.001$) for the *Bubble* strategy with $n = 500$. For $n = 10$, the *Learn New Secrets* strategy worked best, for $n = 50$ and $n = 100$ the *Min secrets* strategy worked best. These strategies had a significant efficiency increase of 12% ($p < 0.001$), 6% ($p < 0.001$) and 3% ($p < 0.001$) respectively. As mentioned in section 2.3, it was expected that the *Random* strategy is a good strategy, which is hard to improve upon. The newly devised strategies, as well as the already existing strategies that were used in this experiment all aim to fan out the agents to which a single agent connects as much as possible. The *Random* strategy apparently already does a good job at this.

4 Conclusion

4.1 Discussion

The strategies in this experiment turn out not to have a big advantage over the *Random* strategy. The best performance for 500 agents was the *Bubble* strategy which had a 3% $\frac{\tau}{\tau_{opt}}$ ratio increase. It is suspected that the top strategies resemble the *Random* strategy too much, as they all have a large element of randomness.

The *Call Me Once* and *Learn New Secrets* strategies use the same underlying principle as the *Random* strategy, which is randomly calling an agent. The difference between these strategies, is that an extra criterion is added to the randomness. As the number of agents increases, the odds that their respective criteria are not met for a specific random agent become increasingly smaller, resulting in a pattern very similar to the *Random* strategy.

The *Bubble* and *Multiply* strategies both try to spread the calls over the agents by using a mathematical formula. This results in a predictable pattern of called agents, until the calculated target agents are already being called. Then, the agents will try to call either a random agent, or the agent next in line, which will then also lead to a pattern similar to the *Random* strategy.

The *Min Secrets* strategy will use information about which agent knows what, to determine the call. All agents that the calling agent has no knowledge about, are presumed to not know any secrets. This results, especially in the beginning, in random calling, since the group of agents which the calling agents has no knowledge of, is very large.

In order to create a strategy that would increase performance even more, one would have to think about characteristics not only based on dividing connections in different time-steps amongst agents. If only dividing connections is considered, as it was in the strategies in this experiment, the strategy is similar to the *Random* strategy and has a similar performance. We can also conclude that the *Random* strategy is already a pretty good strategy, which is very hard to improve upon. Especially the ease of implementing a *Random* strategy and its computationally low cost make the *Random* strategy desirable. When the performance measured in time-steps is however of crucial importance the semi-random strategies pose better alternatives.

4.2 Relevance

The findings in this paper are relevant for systems like Blockchain, where several nodes or agents need to communicate with each other to share knowledge. If, for such a system, a strategy needs to be chosen where communication time must be kept as low as possible, the random strategy should be chosen, because that is the one with almost the lowest completion time, while requiring almost no computational resources, and being very easy to implement. The rest of the strategies needs to have some sort of representation of secrets of others, and/or computation about which agent to connect with. If however communication time is of the utmost importance, the *Bubble* strategy should be chosen.
On the other hand we suspect that there are strategies that would improve the completion time of the random strategy more than 3%, and maybe even approximate the minimal completion time. Different strategies need to be thought of, that are using techniques other than techniques that resemble random picking and dividing connections. This should be determined by future research.

4.3 Team Work

We started this project with a brain storming session, where we thought about the implementation of the project. We together looked up some python frameworks that we could use, and after deciding which ones were most useful, we started on the actual implementation. Roeland and Niek started with the model and controller, where Xabi and Arjan started with the UI. When the basic implementation was done, Niek and Xabi designed and implemented the new strategies for the agents. Roeland later on implemented the simulation version of our program, which was continued by Arjan, so that it could work without the UI. Xabi then did some work to clean up the code, which was thoroughly checked and tested by all members in the group. Arjan mainly continued developing new features in the UI. For the report we again worked on it all together, where different pieces of the report were written, re-read, corrected, re-read and corrected again by different people in our group. Overall, despite the fact that everyone had contributions in both programming and writing the report, Xabi, Arjan were a little more concerned with implementing our program, Roeland with gathering and veryfing the results, where Niek focused on visualizing the results and writing the report.

References

[AGH15] Krzysztof R. Apt, Davide Grossi, and Wiebe van der Hoek. "Epistemic Protocols for Distributed Gossiping". In: *Proceedings Fifteenth Conference on Theoretical Aspects of Rationality and Knowledge, TARK 2015, Carnegie Mellon University, Pittsburgh, USA, June 4-6, 2015.* 2015, pages 51–66. DOI: 10.4204/EPTCS.215.5.

[BS72] Brenda Baker and Robert Shostak. "Gossips and telephones". In: *Discrete Mathematics* 2.3 (1972), pages 191–193. ISSN: 0012-365X. DOI: 10.1016/0012-365X(72)90001-5.

[Coo+19] Martin C. Cooper, Andreas Herzig, Faustine Maffre, Frédéric Maris, and Pierre Régnier. "The epistemic gossip problem". In: *Discrete Mathematics 342 654-663* (2019). DOI: 10.1016/j.disc.2018.10.041.

[DFS14] Benjamin Doerr, Tobias Friedrich, and Thomas Sauerwald. "Quasirandom Rumor Spreading". In: *ACM Trans. Algorithms* 11.2 (Oct. 2014), 9:1–9:35. ISSN: 1549-6325. DOI: 10.1145/2650185.

[Dit+19] Hans van Ditmarsch, Jan van Eijck, Pere Pardo, Rahim Ramezanian, and François Schwarzentruber. "Dynamic Gossip". In: *Bulletin of the Iranian Mathematical Society* 45.3 (June 2019), pages 701–728. ISSN: 1735-8515. DOI: 10.1007/s41980-018-0160-4.

[HMS72] A. Hajnal, E. C. Milner, and E. Szemerédi. "A Cure for the Telephone Disease". In: *Canadian Mathematical Bulletin (447-450* 15.3 (1972). DOI: 10.4153/CMB-1972-081-0.

[Lan54] G. H. Landau. "The distribution of completion times for random communication in a task-oriented group". In: *The bulletin of mathematical biophysics v16 n3 (195409): 187-201* (1954). DOI: 10.1007/BF02478413.

[Net] Mozilla Developer Network. *WebSockets*. https://developer.mozilla.org/en-US/docs/Glossary/WebSockets. Accessed: 2019-10-28.