Model Implementation

Introduction

This section is the main part of the report, which describes the implementation process of the model, consisting of goals, tools, framework building, functional implementation, and adding variables. It is important to note that the NetLogo modeling tool is more than a mere stack of components and images, the dynamic model is implemented by code, so the following section will be accompanied by a presentation of some of the key code so that the reader can better understand the model building process and the implementation of each step.

4.1 Description of objectives

The goal of the model is to simulate the process of data sharing and collaboration and to add protections to the data source to investigate if the protections are effective and if they have an impact on data sharing. To accomplish these goals, the model is divided into two versions, a restricted version and an unrestricted version. The unrestricted version enables data sharing, but that's about it. The restricted version has the option to add protection to the data source as well as to the spy node, and has a line graph that can be used to further investigate the impact of added protection. Therefore, we need to give the assumption before the model is built that applying protection to the data source can be effective in protecting the data, but it can also have an impact on the speed of data sharing. This hypothesis will be demonstrated in the results section of the running test.

4.2 Tool

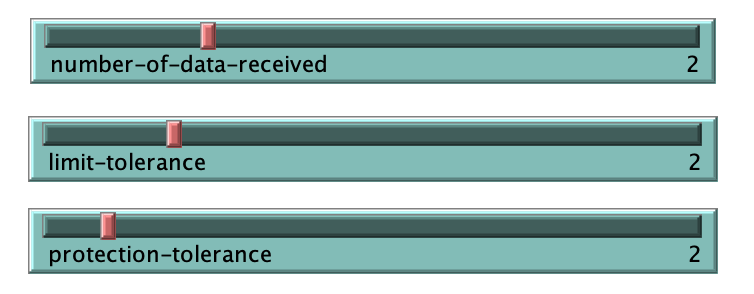
As explained in the literature review section, NetLogo is the modeling tool of choice because of its clear and concise pages and powerful modeling capabilities, as well as its official repository and open source community for beginners to use. In addition, a review of most of the literature on modeling shows that Matlab is used more often for modeling mathematical problems, mechanical engineering, aerospace, and deep learning. This is due to the fact that Matlab is an excellent tool for scientific computation, as all variables are matrix objects, and it uses matrix operations instead of loops, which makes it fast. In addition, Matlab has a simple syntax and is the closest scientific computing language to a general-purpose language. It also supports various language extensions such as python, c, cuda, and so on. However, the disadvantages of Matlab are also obvious, in terms of functionality, its loops are very slow, and the model to be built in this case is designed to simulate real-time changes in data sharing, which requires constant looping of the code (repetitive running of the go button), so using Matlab is very inappropriate and will slow down the running of the model. In addition, Matlab is strictly a software, the whole installation needs 10~20GB, and the running core (similar interpreter) is several hundred meters, which makes the developed program less portable. NetLogo is more like a simulation modeling environment, you only need to download some basic configuration files (107MB) from the official website to use it, and there are no high requirements for the operating system, version, CPU performance and GPU performance. It can be said that after getting a .nlogo model file, the reader can be up and running in a few minutes, saving a lot of time on software download, installation and configuration. Also, NetLogo is very suitable for solving social problems and phenomena, and its own model library has many models based on real social problems, which is very suitable for the theme of this project. In summary, NetLogo's simulation modeling environment will be used for this modeling.

For flowcharting, Microsoft Office Visio was chosen, a software from Microsoft that helps IT and business professionals easily visualize, analyze, and communicate complex information. Visio is the standard software used by most papers, journals, academic journals, etc., so this project will follow that standard.

4.3 functional implementation

4.4 Set variables

This section complements the functional part of the model, as simply implementing the data sharing and collaboration process is only the unrestricted version of the model. Therefore, we also need to set up a restricted version of the model to facilitate the next step, so we need to add some variables to the original model to implement the function of adding restrictions.



As shown in the figure above, we added the Iimit-tolerance and protection-tolerance variables to the model. The Iimit-tolerance is used to impose restrictions on the data source, i.e., after an agent requests access, the Iimit-tolerance variable applies protection to the sharing process of the data source data as a way to secure the data and initially filter out some insecure access sources. This is similar to the firewall settings on Windiws systems, which can be optionally turned off or on, and can perform basic protection functions. However, unlike the Iimit-tolerance variable, which can change its value, i.e., the protection of data sharing is divided into degrees, the larger the value of Iimit-tolerance means that the more tolerant the protection is, the more relaxed the restrictions are, and the easier it is for agents to gain access to the data. Conversely, when we lower the value of Iimit-tolerance, it also means that the data source is more scrutinizing the access request and each agent is less likely to get access to the data, so the protection the data receives is enhanced in disguise. As explained earlier, NetLogo's interactive components are implemented with bound code, and the slider is no exception. Therefore, we can consider the implementation of this feature in the form of a random number, i.e., each time the model executes the 'go' command, each node will be randomized with a value between 0 and 100 to obtain a random number. We then compare this random number with the value of Iimit-tolerance we set, and when the random number is less than the value of Iimit-tolerance, we consider the node to be within the tolerance range of the data source, indicating that the node has gained access to the data. If the random number is greater than the value of Iimit-tolerance, we consider the node to be outside the tolerance range of the data source, marking it as unauthorized and unable to access the data. The next time the 'go' command is executed, the node that did not obtain the data is re-randomized, and the process is repeated thereafter until all nodes have obtained the data.

The number-of-data-received variable is set on top of the Iimit-tolerance variable and represents the amount of data that each data node can acquire (except spy nodes), in order to more recreate the real situation of data sharing and improve the accuracy of the model. Because the reality is that when we share and collaborate on data on the network, we often need only a portion of the data from a large database of data sources, and rarely use all of the data at once. Google Scholar, for example, is a free searchable Google web application for academic articles that includes the vast majority of the world's published academic journals, articles, papers, books, and abstracts, and is an easy way to search a wide range of academic literature. When we access information through Google Docs, we usually search for the literature we want by keyword search, it is impossible to access the Google Docs database directly and then find them one by one, which is like a fantasy. Other data such as images, weather forecasts, etc. are also accessed by accessing one part of the data source to get the information you want. So considering the practical factors, it is necessary to add number-of-data-received variables to the model as part of the simulation. However, it should be noted that due to the uncertainty of the agents represented by each data node, it is almost impossible that some different agents will want to access the exact same data. Therefore, although we set the amount of data that can be fetched by the nodes, the content of the data fetched by each node is random and may vary or be duplicated, depending on the total amount of data from the data source. This also provides a more realistic simulation of data sharing and collaboration.

The protection-tolerance variable is set because of the presence of spy nodes, so this variable indicates how restrictive the data source is with respect to spy nodes. Since the spy node is independent of other different nodes and data source nodes, it represents uncertainty. The higher the value of protection-tolerance, the weaker the data source is against spies or criminals, and the more likely it is that a spy will succeed in breaking through the data source's protection. Conversely, a smaller value of protection-tolerance means that the data source is less tolerant of criminals, and these spy nodes will be denied access to the data if they are not careful enough to steal it. But it is worth noting that, even if the spy node lucky to break through the protection mechanism of the data source, is not necessarily to get the data, because the actual situation, the server or agent will be important data or data privacy involved in setting more stringent protection measures. This means that it is very difficult for a spy to get all the data of a server. To simulate this situation, we set a private variable for the spy node, denoted by a random number. Each time the model executes the 'go' command, the spy node will represent the total amount of data stolen by the spy node as a random number in the range of 0 - the total amount of data held by the data source (including both 0 and all data). This simulates the uncertainty of the data stealing process. The reality is that attacks on servers and data sources do not always succeed, even at the risk of committing a crime and being arrested, because people are becoming more aware of data security and data protection in recent years.

In addition, in reality, we may also encounter situations in which we receive sales advertisements, insurance advertisements or scam messages from salespeople, insurance companies or criminals who do not have direct access to our personal information and to whom we have never provided it. This is because these salespeople, insurance companies or criminals get some of our information, such as name, age, etc., from other organizations or platforms, and some other information, such as address, phone number, etc., from some platforms. It is through this kind of data reconstruction that these annoying organizations or individuals have access to all of our personal information that we have made public. In fact, this is no different from committing a crime, as it is illegal to obtain privacy and harass people. Illegal reconstruction of data is an impossible to prevent, because the service provider or data source can only authenticate and limit the first level of agents, but there is no direct link to the second, third or lower level of agents. Therefore, it is not possible to effectively protect and restrict the subsequent data dissemination process. The model also attempts to solve this problem, and we distinguish between two cases when setting up a spy node: the spy node is directly connected to the data source; the spy node disguises itself as a Level 2 or Level 3 agent and is not directly connected to the data source. The first case, as described above, is controlled by the protection-tolerance variable. The second case is the one we are currently discussing, where the source data is potentially reconstructed by a low-level agent. X represents the total amount of data (excluding duplicates) held by all nodes directly connected to the spy node, which means that the spy node does not always succeed in Reconstructing the source data, as the data used by its higher-level agents may not be all the data, depending on our total number of nodes, number-of-data-received values and other data nodes each time random results. By running the model multiple times we find that with a total number of nodes of 10 and number-of-data-received of 3, the probability of the spy node acquiring different amounts of data is shown in the following figure.

From the above diagram, we can easily see that the probability of the spy node successfully reconstructing the data (getting all five data) is very low, which means that the protection measures we have set up are very effective, and to some extent, they solve the problem of the source data being maliciously reconstructed, which further protects the security of the data.

Finally, to facilitate further study of the model later, we added line diagrams to the model. The horizontal axis of the line graph shows the runtime of the model in 'tick'. It should be noted that in NetLogo, if the model is dynamic, it is represented as '1 tick' every time the model is executed, and we usually use 'tick' to represent the runtime of the model. This way, if we set up a log when a runtime error occurs, we can see exactly which 'tick' the error occurred at, which makes it easy to modify and debug the code. The vertical axis of the line graph represents the percentage of the total number of nodes, in %. Thus the line graph shows the variation in the number of different types of nodes as a function of time/number of runs. The line graph can provide us with data for quantitative analysis in the subsequent hypothesis validation section to get more accurate conclusions.

At this point, all the features of the model including variables have been added and functionally divided into two versions. The next section will provide a detailed description of the testing and running process of the model.