2018

CS310 – Progress Report

NORM EMERGENCE WITH FORGIVENESS

JONATHAN GOH (1602669)

Introduction

Social norms are patterns of behaviour that exist within a system where there is no central authority. The reason for a system such as this to exist range from privacy requirements to the efficiency of a system. As a result, these norms are needed for the individual entities to communicate with each other in a cooperative manner. These patterns are promoted through a mechanism called metanorms which dictates that one must punish those that do not follow the norm. The aim of this project is to investigate the effectiveness of metanorms in different topological systems and to introduce a forgiveness mechanism to the model.

Whilst Axelrod's model [2] shows how norms can be established within a population, it assumes that the network is complete, with all entities able to observe all other entities in the network. However, in real world applications, this may not always be the case and clusters of nodes may appear instead, with fewer connections to other clusters throughout the network as well as each node having much fewer connections. Systems like these can be seen everywhere, for example a university may have many nodes in its network, but not all students will have a connection with everyone else in that network. Instead, the students will form clusters of groups due to the characteristics of each person such as attending the same course or being a part of the same societies, and each cluster of students will have connections to other clusters in the network randomly.

The project will also introduce a new forgiveness model whereby observers that see an agent that does not comply with the norm in the system will also be given an option to forgive the act. In the real world, this mechanism may reflect someone who has committed an act of defiance but is not severe enough to receive a harsh punishment from an observer. Within the project, the way this interaction works has yet to be confirmed, however the general idea is that the agent that doesn't comply ends up paying a small to no cost for that interaction.

The GitHub for this project: https://github.com/JonoGoh/cs310

Objectives

The aim of this project is to investigate the effect of forgiveness within different network topologies. The first part of the project involves the replication of the models generated in Mahmoud, Griffiths, Keppens and Luck's work [1] and once that has been achieved, to add another forgiveness mechanism to their model. The requirements for the project are as follows.

1. Each agent has a value for *boldness* and *vengefulness*
    - This value is how likely the agent is to defy the norm and how likely the agent is to punish an agent that has not followed the norm
    - Values will change over the iterations of the model
2. Agents must be able to interact with each other properly.
    - Each entity must choose whether to comply to the norm or not.
    - If an agent does not comply to the norm, then all observing entities will have a choice whether to punish that agent with large cost to the defect and little cost to itself.
    - An observer that does not punish the agent not complying to the norm may potentially receive a punishment themselves (metapunishment) from a second observer observing the situation.
3. A forgiveness mechanism or many forgiveness mechanisms have been introduced to the model and tested.

    - Observer may choose to forgive the defective agent with little or no cost to both involved. This may be based off the history of an agent, such as whether the agent has a good track record of not defecting, or on a new forgiveness(empathy/mercy?) variable, like boldness or vengefulness.
4. Model must be able to work over many topologies.
5. Results from the model should be output in graphical form such that the characteristics of the agents/network over time can be viewed clearly such that clear conclusions can be drawn from each variation of the model.

Current state of the Project

The project is currently being written completely in Python, using both Atom and Jupyter to edit the code; Atom is used to develop the model mechanics and Jupyter is used mainly to analyse the data from the models generated. NetworkX is the main package being used to generate the networks and create the models, with some use of Matplotlib and Numpy to analyse the model.

Objective progression

Below is the current progress towards each of the requirements outlined int Objectives section:

1. Each node in the network is relabelled with a unique agent class after the graph has been generated. The agent class contains variables for the boldness, vengefulness and defection, punishment, punishment omission and total scores as well as functions that reset and total the scores, generate random values for the boldness and vengefulness when the agent explores and print information about itself for debugging. Each agent is created at the start of the model with randomly generated value between 0 and 1 (with a step of 1/7) meaning there are 8 possible values for the variables to take.
2. The Agents can interact with its neighbours properly as explained in the requirements.
3. No work has been done on this requirement. However, the next two weeks of term will be used to plan out different ways to implement this mechanism so that I will have a good idea for what I need to do for next term.
4. The model works over a range of topologies, although this has yet to be documented fully. An extensive breakdown of how agents act over different topologies will be included within the final report including the reaction of the forgiveness mechanism over these topologies.
5. Graphical outputs of the model can be generated as well as other graphical forms that show the development of the agents over time.

Issues during development

Several smaller issues have presented themselves during the development process of this project. While nothing major has happened so far, precautions will still need to be taken to prevent and mitigate risks (outlined in the risk section of the paper).

The main issue that occurred during the project so far was redesigning the way the model worked. Initially, the model used the attributes of each node, from the default NetworkX node class, to list its characteristics such as boldness, vengefulness and its scores during the game. However, as the model became increasingly complex, using the node attributes became very inefficient when writing the code and running the model. To fix this, an agent class was created which contained information about each agent, as well as some useful functions used to reset the scores and to debug the code. All the nodes are then relabelled with the agent class using the relabel_nodes() function after the graph was generated.

Another issue faced with during the project is the overrunning of tasks/tasks taking longer than expected, which causes other tasks to become delayed. While most of these delays are due to unforeseen circumstances, more could be done to better manage the project schedule. As a result of these delays, the way in which forgiveness will be implemented into the model has yet to be decided, and the next 2 weeks after the progress report is handed in will be used to properly plan this mechanism and the schedule for the next term. I have altered the timetable slightly to a accommodate for these delays, as well as using a new scheduling software to more effectively organise the project.

Testing

The testing involved in this project is different from the way most other projects will be tested. Instead of testing different components to see whether the outcomes of the tests meet the expected values, there will not be any expected values for this project since we are unsure of how the new forgiveness mechanism(s) will affect the outcome of the model. Things that may be tested in the project include:

- How the mechanism affects the model overall
- How the mechanism may react over different topologies
- How changing gain and cost values affects the model

Preliminary results.

I have been able to replicate some of the results within Griffith's[1] and Axelrod's[2] paper, specifically applying the model to complete graphs and to some lattice topologies, where the neighbourhood size is changed. These results are the same as the ones found in the paper and so will not be repeated here. I am currently working on some optimisations to the code to make it run more efficiently without changing the underlying mechanisms of the model.

Methods

For this project a mixture of plan-driven and agile methodologies will be used. Whilst a plan and a timetable for the progress of the project has been created for the many weeks ahead, there is a very high chance that these plans and requirements may change over the course of the project, meaning the time line must adapt to fit accordingly. A plan driven approach to this project is also useful since there are hard deadlines submit the various deliverables and having a structure for the project will greatly help achieve these goals. This is unchanged from the time the specification was written.

Timetable

The timetable has been outlined on the last page and has been changed since the specification due to reasons highlighted in the issues section of this report. The new timetable has some tasks moved around slightly to accommodate for some of the delays and has been moved to another platform that allows me to better track the progress of the project.

Meetings with my supervisor have been running and will continue to run every week where we will discuss the progress and direction of the project. These sessions are not scheduled within the timetable itself, and all the meetings have been recorded on the tabula website.

Resources

The following are a list of resources that are currently being used within the project:

- Python – The language the program will be written in. This is the chosen language as there are many well documented packages that are available to use that are relevant for this project.
- Git/GitHub – Create and store backups of my work so that the code can be accessed from different computers as well as access earlier versions of the code in case something happens to the current version
- NetworkX/graph-tool – Creating networks to use for the models
- Plotly – Output of networks graphs to more clearly see the interactions between entities. This package generates graphs in a much more aesthetic fashion so may be useful for the presentation during the second semester.

New resources that are being used since the specification was written:

- Atom – Modern text editor preferred due to the many packages that can be installed to customise the experience.
- Matplotlib [4] – Also a package that outputs graphs in a visual form. May also use this software to generate other graphs that show what happens in the models. This package is much simpler than Plotly to generate the graphs and so has been used in place of plotly so far to visualise the model.
- iPython Notebook/Jupyter [6] – Having used this environment for the CS342 coursework, it provides a clear work environment from which chunks of code can be executed independently and graphs can be output in line with the code.
- TeamGantt – A free online scheduling software that allows me to plan and manage my project efficiently.

Risks

The following are a list of risks that may be encountered during the project.

- Something happening to the current version of the code such as losing the code or files getting corrupted. To prevent this from having a big effect on the project, frequent backups will be made and stored on GitHub
- A python package that is being used malfunctioning with the code. This could be due to an update for the package that results in compatibility issues, or a server going down for an API in use. To remedy this, python packages that are needed will be downloaded and stored locally so that if there is an update that breaks the program, then going back to an earlier working version of the package may be the solution.
- As the networks become more complicated and the interactions between the nodes become more complex, the program may become much more difficult to manage and possible even run slowly, hindering progress. The best way to deal with this is to lay out my work clearly and to properly document the project throughout, as well as maintaining proper coding principles.
- The packages used may not support the networks or the network functions that are required for this project. If this happens then another package that meets the requirements will need to be used or, in the worst-case scenario, create these network functions from scratch.

Risk 3 is the only risk that has happened so far and has caused the project to become delayed. This is in part due to poor planning during the design phase of this project and has caused a complete redesign of the model. The risk has now been taken care of and more will be done to prevent further delays in the project.

Legal, Social and Ethical considerations

Since the project does not involve the use of any personal data, there are no legal, ethical or social considerations that are relevant. This has not changed since the project specification.

Conclusion

Overall the project has progressed well over the past 8 weeks. Most of the requirements set within the specification have been met and while the project has faced a few issues, these were eventually remedied with solutions that will allow the project to progress a lot more efficiently and have not completely hindered progress. As for what needs to be done for the future, a better idea of how the forgiveness mechanism will work will be planned for the next two weeks as well as testing of the model over different topologies. For next term, the implementation and testing of this mechanism will be completed as well as the final report and the project presentation. Overall, despite the delays encountered, I am on track to complete the project within the deadline.

References

1. Samhar Mahmoud, Nathan Griffiths, Jeroen Keppens, Michael Luck, *Establishing Norms with Metanorms over Interaction Topologies*, available at https://www.dcs.warwick.ac.uk/~nathan/resources/Publications/jaamas-2017-author-version.pdf
2. Axelrod, R. (1986). An Evolutionary Approach to Norms. American Political Science Review, 80(4), 1095-1111. doi:10.1017/S0003055400185016
3. Networkx.github.io. (2018). *Reference — NetworkX 2.2 documentation*. [online] Available at: https://networkx.github.io/documentation/stable/reference/index.html [Accessed 26 Nov. 2018].
4. Matplotlib.org. (2018). *User's Guide — Matplotlib 3.0.2 documentation*. [online] Available at: https://matplotlib.org/users/index.html [Accessed 26 Nov. 2018].
5. Jupyter.readthedocs.io. (2018). *Overview — Jupyter Documentation 4.1.1 alpha documentation*. [online] Available at: https://jupyter.readthedocs.io/en/latest/ [Accessed 26 Nov. 2018].