Network Software Modelling Assignment 2

James McDermott

February 18, 2017

This is a group assignment (groups may consist of 1, 2, or 3 students) worth 25% of the module. It is

due 23:59 Sun 23 April (end of Week 11). The topic of this assignment is information-flow simulations in

different types of graphs.

Your submission must consist of 1 pdf of up to 5 pages (minimum 11 point type) plus one Python file.

This is short so words, tables, and figures should be chosen carefully. If the Python file needs to read a graph

from disk, the file should be included also; or it may read a graph from a URL. Your pdf and Python file

should each carry all group members' names and ID numbers at the top.

You are free to discuss this assignment with other students, but you may not show your group's work

to another group or look at another group's work. You are free to use external resources if you cite them

correctly. However, the large majority of your work must be written by you, in your own words, as a direct

answer to the questions of the assignment, rather than quoted or paraphrased (even with citation) from

external resources.

In this assignment, we will think of the nodes of a graph as agents, such as people, organisms, companies,

or software entities, which communicate in some way. Communication happens only via edges. For example,

a directed edge from A to B may indicate that B follows A on Twitter, hence receives information from

A. The same information may later propagate to those who follow B. For another example, in a field of fireflies, “information" is passed whenever one firefly sees a neighbour flashing. It seems that the whole field

of fireflies often manage to synchronize their flashing despite each individual only seeing a few neighbours.

In simulations such as this, we often think of each node (agent) as being in a “state"; iteratively updating

this state in response to incoming messages; and sending out messages in response to its current state and

any incoming messages. For example, an agent representing a Twitter user may be in “reading" or “writing"

states. When reading it may send retweets, in proportion to the number of messages it receives, and when

writing it may create original messages. An agent might transition from reading to writing if it does not

receive any messages for several time-steps. When studying information ow we may represent messages

by simple data, such as integers, rather than simulating the details of the Twitter messages themselves. In

some simulations, the graph itself may grow or change over time. In summary, a simulation has simulation

rules which correspond to the real-world phenomenon being modelled.

We can use simulations to ask questions such as: Does information eventually reach all agents? If so, how

long does it take? Do some agents consistently receive it sooner? Are some agents crucial to information

flow, or are all agents equally important? Does the behaviour of agents synchronize, or converge, or do they

tend to behave independently? Where there are competing messages, does one tend to take over the whole

population, or do they settle into equilibrium, or is there an oscillation-type behaviour? We will call these

simulation properties.

For any of these questions, we can further investigate: Does the answer depend on the type of graph

(e.g. one of our random graph models, a regular graph, a real-world graph), or on the number of nodes, the

density of edges, or something else? We will call these graph properties.

Report outline

Tasks:

1. Describe briefly the real-world phenomenon you wish to model.

War is the subject that forms the basis for our simulation modelling.

If we think about war, it has the properties of infection – as countries are taken over, they can be considered infected.

A country does not ‘skip’ a country, rather it attacks a country adjacent to an allied one. Once conflict is resolved, the attacking country is either successful and takes over the territory (it is infected) or the defending country successfully repels the attack and remains in the state it was in before the attack uninfected).

If, for example, we consider WWII, Germany initially attacked Poland, in order to be in a place where they could attack the Soviet Union from. They had to capture the adjacent country (node) in order to progress to the following one, as ultimately, they were interested in capturing the Soviet land, and not the Polish land.

The same principles hold true today. While the nature of war has changed since the introduction of manned/unmanned aircraft and more sophisticated marine-based vessels such as aircraft carriers (both of which make it more difficult to model real-world war), the fundamentals still imply that there must be forces that occupy the ground for it to be considered taken over.

If we represent each country with a node, where edges connect it to its adjacent neighbours, we can consider the world a network graph. Alternatively, if a country has no adjacent countries (e.g. Iceland), we may consider the countries closest to it as the nodes it in its neighbourhood, and connected to it. For example, the world may be viewed as the below graphs show:

Code:

We want to emulate the scenario whereby if 3 factions were to go to war, who would come out on top based on several different factors, which WI contribute towards an attack & defence score at each timestep:

*GDP* – This is significant as the higher the GDP, the greater the financial might of the country and the greater the country’s ability to finance war.

*Population* – The army will need to consist of enough people to fight the war. The more people, the higher the likelihood of having a talented, well-trained, motivated army at the country’s disposal.

*Military Spending* – The historical Military Spending of a country is a good indicator of the country’s attitude to war. It will also be a good indicator of the military experience a country will have.

*Distance from Home* – It can be assumed that the further away a battle takes place from the home ‘base’ of a country, the tougher it will be for a country to succeed in capturing another country (e.g. USA in Vietnam)

*Number of different fronts* – The more fronts each faction fights on spreads the forces thinner and so will reduce the number of soldiers dedicated to each front, reducing the chances of military success in battle.

*Random Multiplier* - In any conflict, there will be elements of luck that will impact both sides. This factor is included to replicate this luck factor.

2. Describe briefly your choice of graph model and how it corresponds to the real-world phenomenon,

e.g. your choice of directed versus undirected edges, edge weights, allowing or disallowing self-loops,

etc.

3. Simulation Rules

The simulation starts out with 3 countries representing 3 factions – for example, we’ll take the USA, China and Australia as the 3 factions.

For each territory (country) under the control of each faction, the country will attack a neighbouring territory if either (a) it is under the control of rival faction or (b) it is not owned by the attacking faction or a rival faction (i.e. it has yet to be taken over by anyone).

The order can be simulated or done in a cycle – it shouldn’t be a breain

We find the information pertaining to each factor above through online sources. *GDP, Population, Military Spending* are all easily sourced from the internet. *Distance from Home* is calculated per conflict, as is *Number of different fronts* and the *Random Multiplier.*

A crude model may represent the outcome of a conflict between factions. Such a model may be:

Conflict\_Score = ; where the Random Multiplier is between 0 and 1.

The conflicts are resolved one at a time, in each time-step. If a faction is attacking and has a higher conflict score than the territory it is attacking, it is successful and the country being attacked is ‘infected’ and added to the attacking faction’s portfolio of countries under its control, and is removed from the defending faction’s portfolio . Alternatively, if the attacking faction has a lower conflict score than that of the defending country, no change takes place to either faction portfolio.

2. Describe briefly your choice of graph model and how it corresponds to the real-world phenomenon,

e.g. your choice of directed versus undirected edges, edge weights, allowing or disallowing self-loops,

etc.

3. With the aid of a diagram, describe the simulation rules, e.g. how agents change state and what

causes them to send messages.

4. Program your simulation in Python. You may use NetworkX, Pregel, Numpy, Scipy, Pandas, Mat-

plotlib, Seaborn, Statsmodels, and the Python standard library. You may also use other libraries for

optimisation, statistics, visualisation, and so on. But if you wish to use any unusual libraries which

relate to graphs or simulation, please contact me first to confirm they are allowed. The main rule will

be that you are required to write your own graph simulation, not to import one written by someone

else.

5. State one or more simulation properties which you wish to investigate and which graph properties

you hypothesize they may depend on.

6. Carry out experiments to measure these properties in different types and sizes of graph (at least one

real-world graph, and at least one scalable graph model such as the Erdos-Renyi random graph model,

at multiple sizes). Present a table of data showing your results.

7. State your conclusions, based on your data.

Marks will be awarded for: an interesting and original real-world phenomenon; an accurate modelling of

it; good-quality, well-commented, efficient code; an interesting hypothesis/research question; understandable

results which go towards answering the question; firm conclusions; a well-written document with good

diagrams, tables and figures, as appropriate.