# 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 flow 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**.

Tasks:

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