Study guide

The purpose of this session is to learn about how to simulate changes in state on the nodes of a network. Many networks have a static structure (the nodes and edges are unchanged), but with changing state on the nodes or edges. We explore dynamic state on networks using the SIS model of epidemics.

In order to be ready for class, you need to be able to do the following.

- Use Python dictionaries to add and modify properties on nodes and edges in a graph.
- Simulate changes in state on the nodes and edges of a graph.
- Understand how we replace synchronous/sweep updates with randomized asynchronous single-node or single-edge updates.
- Visualize networks using `networkx`.

Storing state on networks

- We might want to store state on the nodes or edges of a network. Using NetworkX this is done by assigning properties to nodes or edges as if they were Python dictionaries. Example:

  ```
  g.nodes[i]['age'] = 23
  g.nodes[i]['nationality'] = 'Burkinabe'
  g.edges[i, j]['relationship'] = 'parent'
  ```

  Note that `i` and `j` are the labels of two nodes. These variables could contain strings or integers or any other immutable Python type.

- Remember to use the code for Chapter 16 available from this repository instead of the code in the textbook.

Models from the textbook

The *majority* example (pp 327–330) shows how to store the visual layout of a graph so that you can reuse it later.

- This is handy for keeping the visualization stable rather than doing a different layout as the graph changes (or making a new random layout every time).
- The discussion about memory efficiency and iterators on p 327 is irrelevant when using NetworkX 2.0 or above since the library now uses iterators by default when accessing the `nodes` property of a network.

The *voter* example (pp 331–333) introduces asynchronous updating.

- Before we updated the state of all nodes and edges synchronously (in one sweep using a for loop).
- With asynchronous updating, a random node or edge is selected and updated according to the rules of the model. This removes one or more for loops from our update method.
- But the meaning of “one step” is now different–
  - With synchronous updating, all nodes and edges were updated in one step.
  - With asynchronous updating only one node or edge is updated in one step – so expect more steps.

We explore the *epidemic* example (pp 333–335) further in class, but you should still read through it before class to understand how the network dynamics attempt to model the spread of disease in a population.

- Finding the critical values at which an epidemic does or does not spread links back to the *mean-field approximation* and *renormalization group analysis* concepts from the cellular automata unit.

You can skip continuous state and time models – pp 335–347. This is advanced reading if you are familiar with differential equations and Euler’s method for approximating their time evolution.