Dynamic Processes on Graphs

Francesco Giuseppe Gillio
Politecnico di Torino

I. PROBLEM OVERVIEW

The research attempts to simulate dynamic processes on graphs to study the effect of initial conditions on dynamics. The simulator attempts to efficiently generate G(n,p) graphs with 10k nodes (for 100k nodes the simulator exceeds the computer's operating limit). Furthermore, the simulator attempts to establish a regime of discrete events according to a Poisson process with lambda intensity equal to 1 to simulate discrete 'node wakes up' events.

II. PROBLEM SOLVING

In the current section the report attempts to address, step by step, the different layers of the problem solving approach, such as the data structure, the FES managemen and the G(n,p) graph sample generation algorithm.

A. Data structure

The search implements the simulator according to the following data structure:

- the `Node` class, set to contain the node state and the set of connecting nodes, and to update the node state according to a majority model.
- the `nodes` dictionary, which records nodes
- the `generation` function, which generates the graph G(n,p) according to the number of nodes N and the probability P.
- the `model` function, which triggers the majority model for each graph node when the wakes up event occurs.
- the `simulation` function, which establishes the event loop regime, with time as the event loop termination criterion.

B. FES managemen

The FES management occurs in accordance with previous simulation architectures. The FES therefore archives a succession of wake up events, each of which establishes the future of the following one according to a Poisson process with lambda equal to 1. Upon extraction of the wake up event, the simulator operates the model's algorithm majority at each node of the graph.

C. G(n,p) graph sample generation algorithm

The generation of the G(n,p) graph occurs according to the generation function of the simulator. The function generates n nodes, each with a given state equal to 0 (off) or 1 (on). The simulator assigns a state to the node randomly, according to a probability distribution that favors the off state (the simulator initializes approximately 75% of the nodes with the off state). Then, the generator creates links between the nodes with probability p.

III. RESULTS

The simulator results show how the majority model increases the inequality of the initial states. The graphs show some solutions for different values of the simulation time limit.

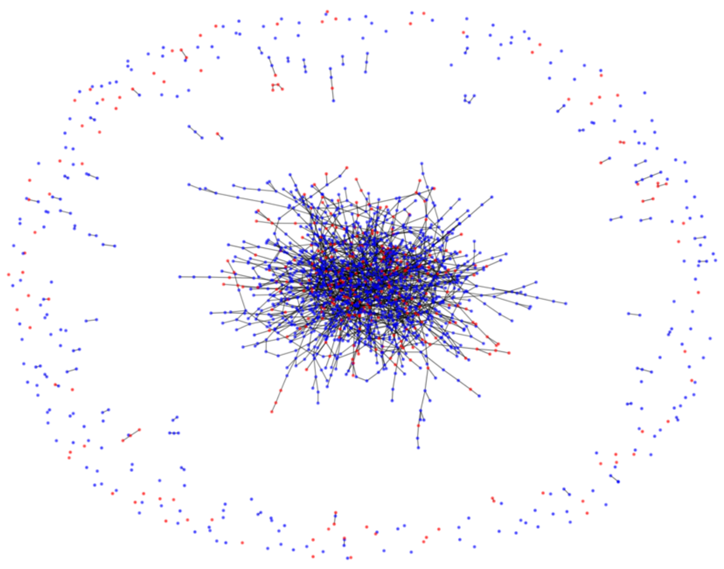

Figure 1 - on: 260, off: 740

The simulator generates a graph G(n,p) with the initial distribution of nodes shown in Figure 1, where the blue color represents the off state of the node, while the red color represents the on state.

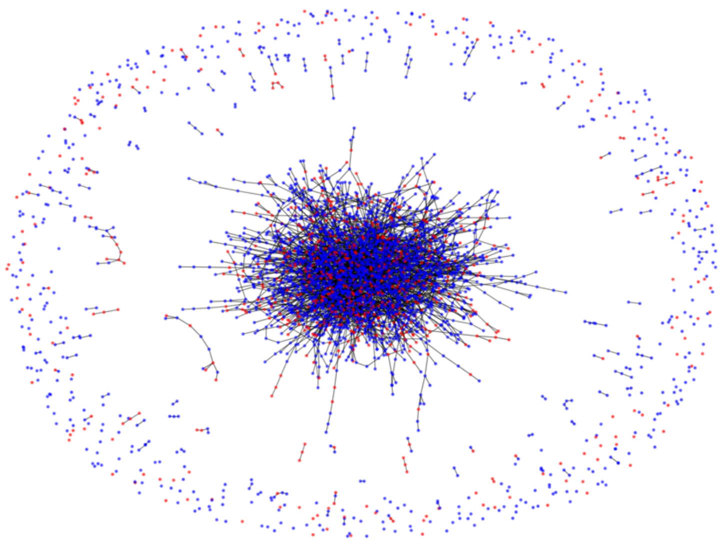

Figure 2 - on: 170, off: 830

Figure 2 shows the situation of the node states after 10 iterations, therefore after the advent of 10 wake up events.

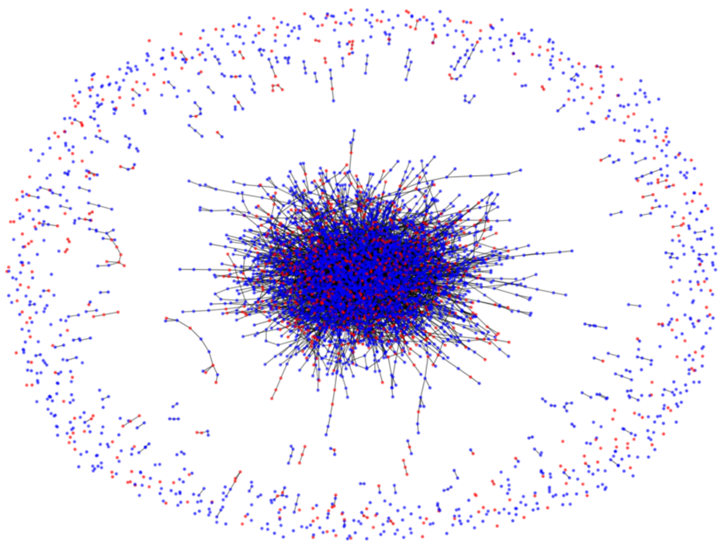

Figure 3 - on 135, off: 865

Figure 3 shows the situation of the node states after 100 iterations, therefore after the advent of 100 wake up events.

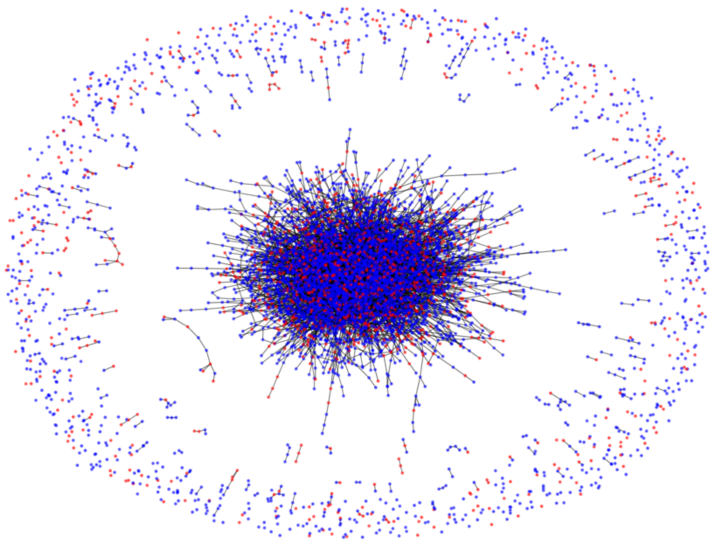

Figure 4 - on: 125, off: 875

Figure 4 shows the situation of the node states after 1000 iterations, therefore after the advent of 1000 wake up events.

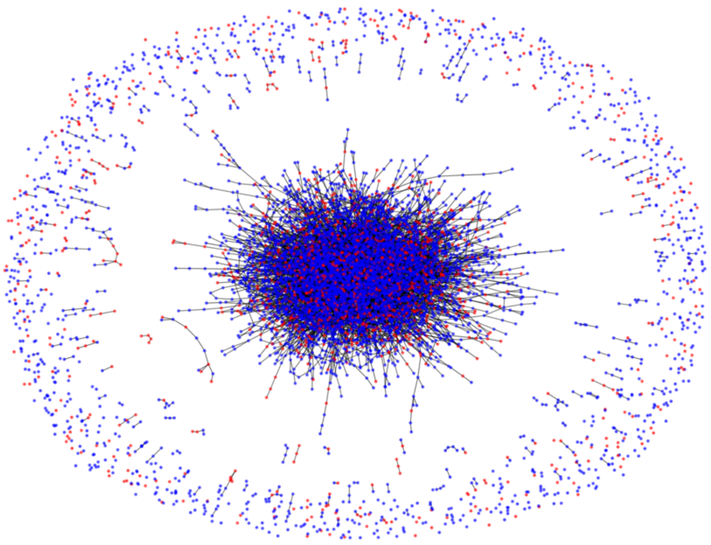

Figure 5 - on: 120, off: 880

Figure 5 shows the situation of the node states after 1000 iterations, therefore after the advent of 1000 wake up events.