Complex Networks

Exercise 3

Author: Aleix Solanes Font

Course: Fall semester 2015/16

Table of Contents

About the code

All the code is found in the file SolanesAleix-sis_epidemic.py.

The code is developed in python, an interpreted language. As this simulation might take many time to finish (the best choice would have been to implement it in a compiled language like C), I have used an alternative implementation of the Python module, which is called PyPy.

PyPy is a fast, compliant alternative implementation of the Python language that takes advantage of a Just-in-Time compiler which does a compilation during the execution of the program. This makes python code usually to run the code faster that simply executing it under the Python module by itself.

As an idea of how the code improved the speed, in my code, running it under Python 2.7 took more than 15h on an Intel i5-6200u processor, but running the same code under PyPy took about 7h to complete the full execution.

As a library to implement the different networks, the python library **Networkx** has been chosen because of its simplicity and the previous knowledge I had to develop fast by using it.

In order to plot all the results I used the **matplotlib** library, which in this case saves every plot into a .png file to be able to attach all these plots easily to a word document like this one.

The implementation

In this practical work, I have used three different kind of networks:

- Erdös-Rényi
- Barabasi Albert
- Random network

For each network, I have used both a 500 and a 1000 edge model.

μ: Spontaneous recovery probability. I have tested 3 different values (0.1, 0.5, 0.9) in order to cover the full spectrum from a very low probability to a high degree probability.

β: Infection probability of a susceptible individual when it is contacted by an infected one. Like proposed, I have used 51 values and an increase of 0.02.

N_{rep}: The number of repetitions has been set to 100.

p(0): the probability of each node being initially infected has been set to 0.2.

T_{max}: Simulations of 1000 time steps.

T_{trans}: The number of steps of the transitory has been set to 900. So, the stationary is formed by 100 steps (1000-900).

Results

Erdös-Rényi

500 edges

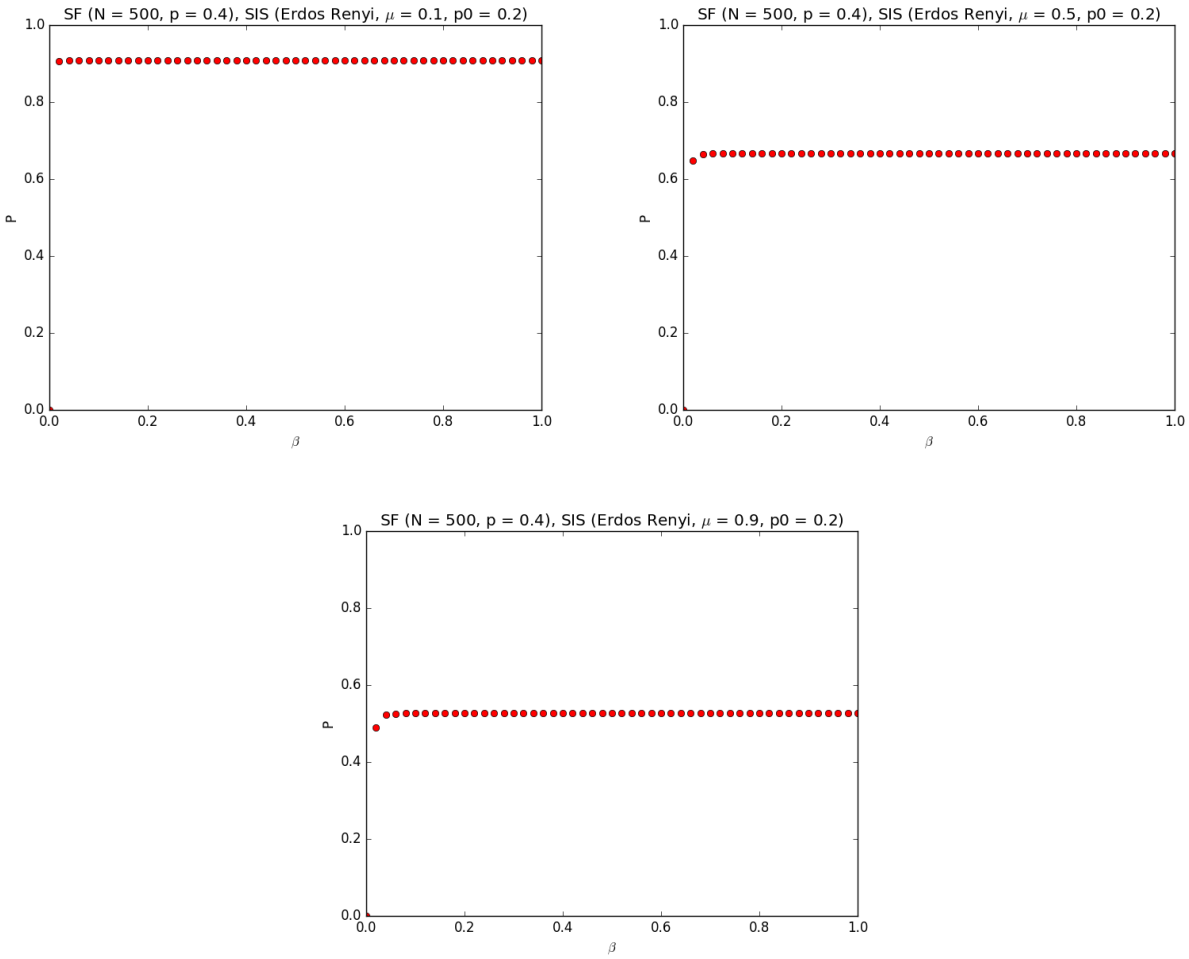
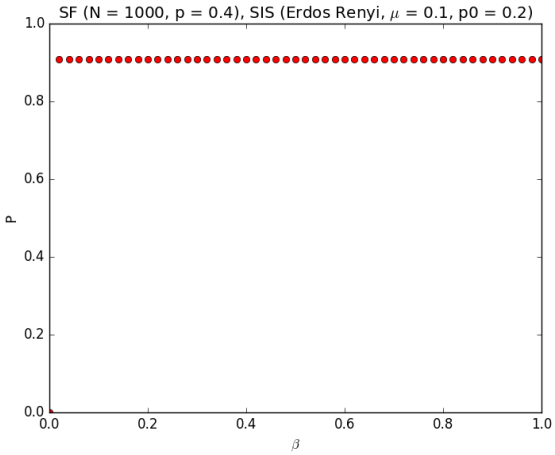
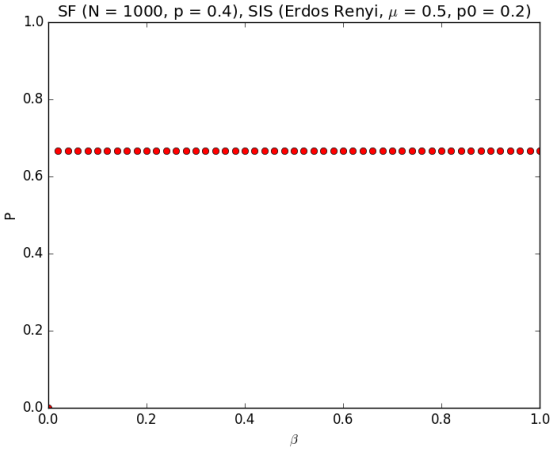


Figure 1: Plots of Erdös-Rényi 500 edges. P as a function of Beta.

1000 edges





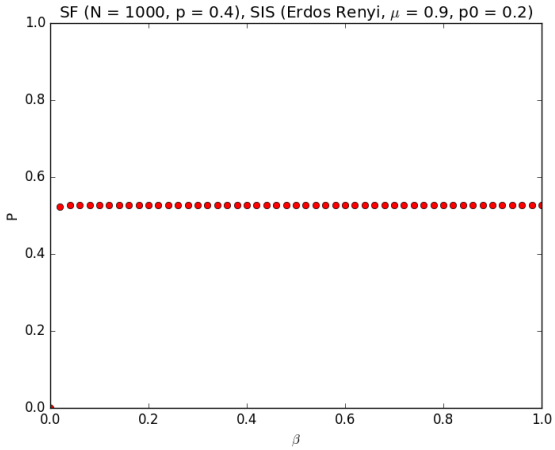
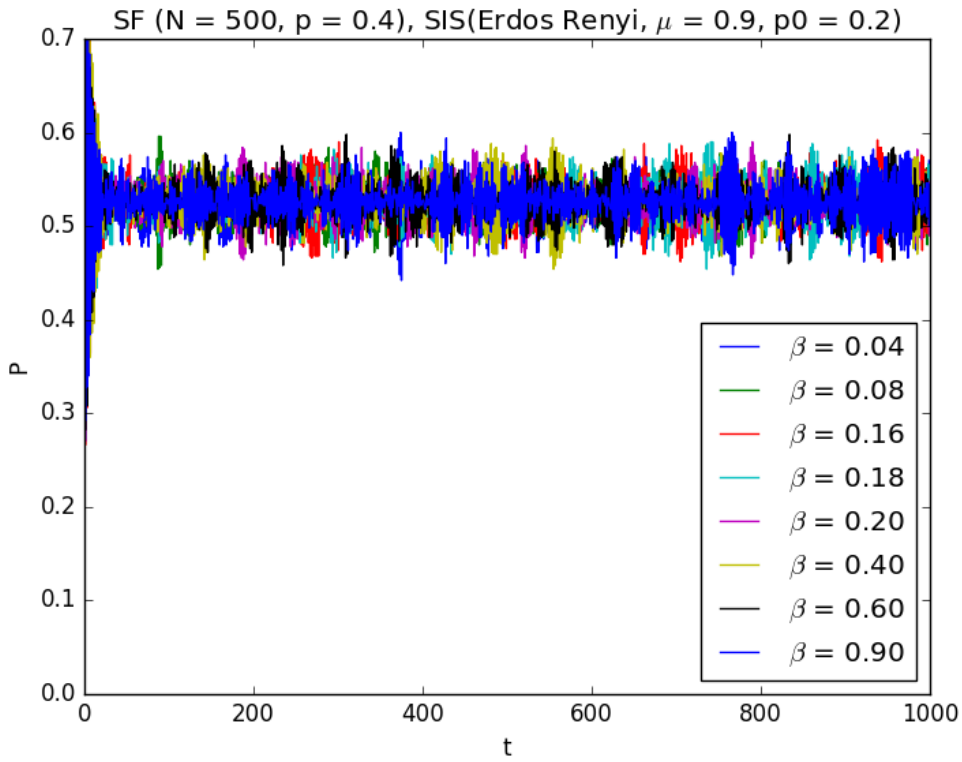
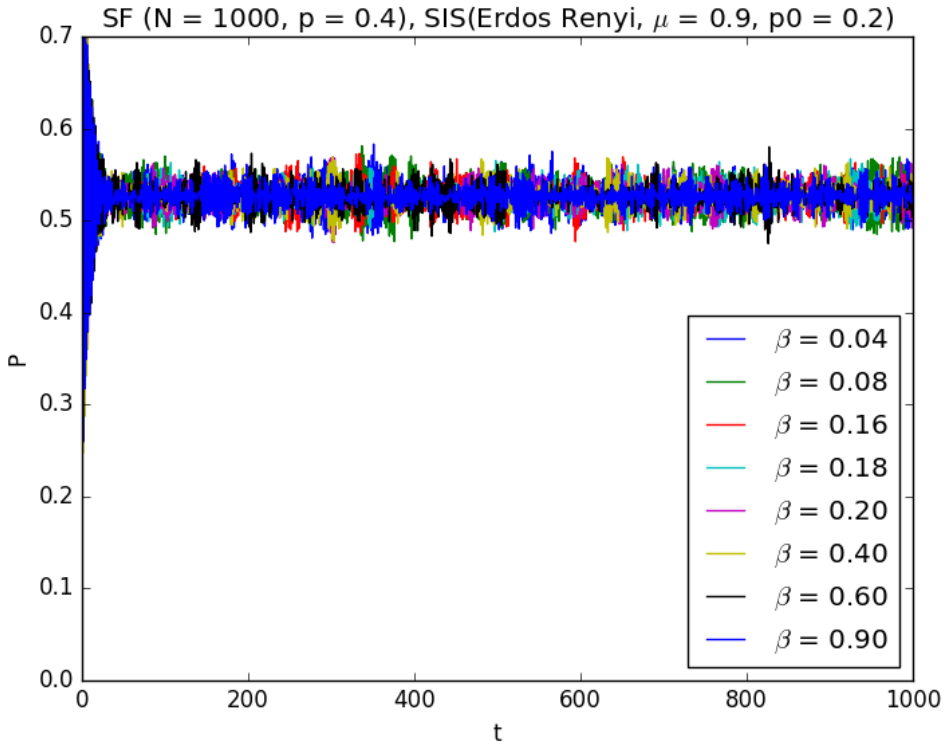


Figure 2: Plots of Erdös-Rényi 1000 edges. P as a function of Beta.

Simulation plot



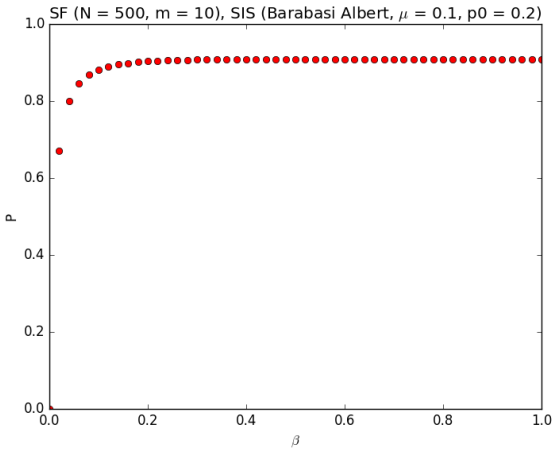
Simulation plot: Erdös-Rényi 500 edges

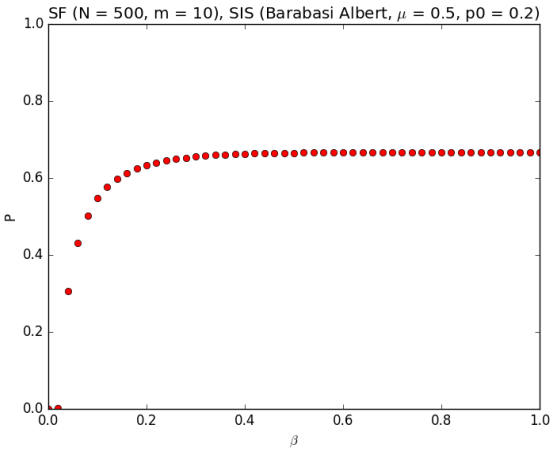


Simulation plot: Erdös-Rényi 1000 edges.

Barabasi Albert

500 edges





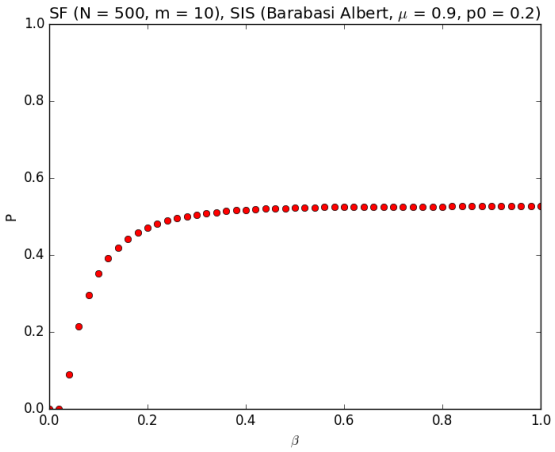


Figure 3: Plots of Barabasi Albert 5 edges. P as a function of Beta.

1000 edges

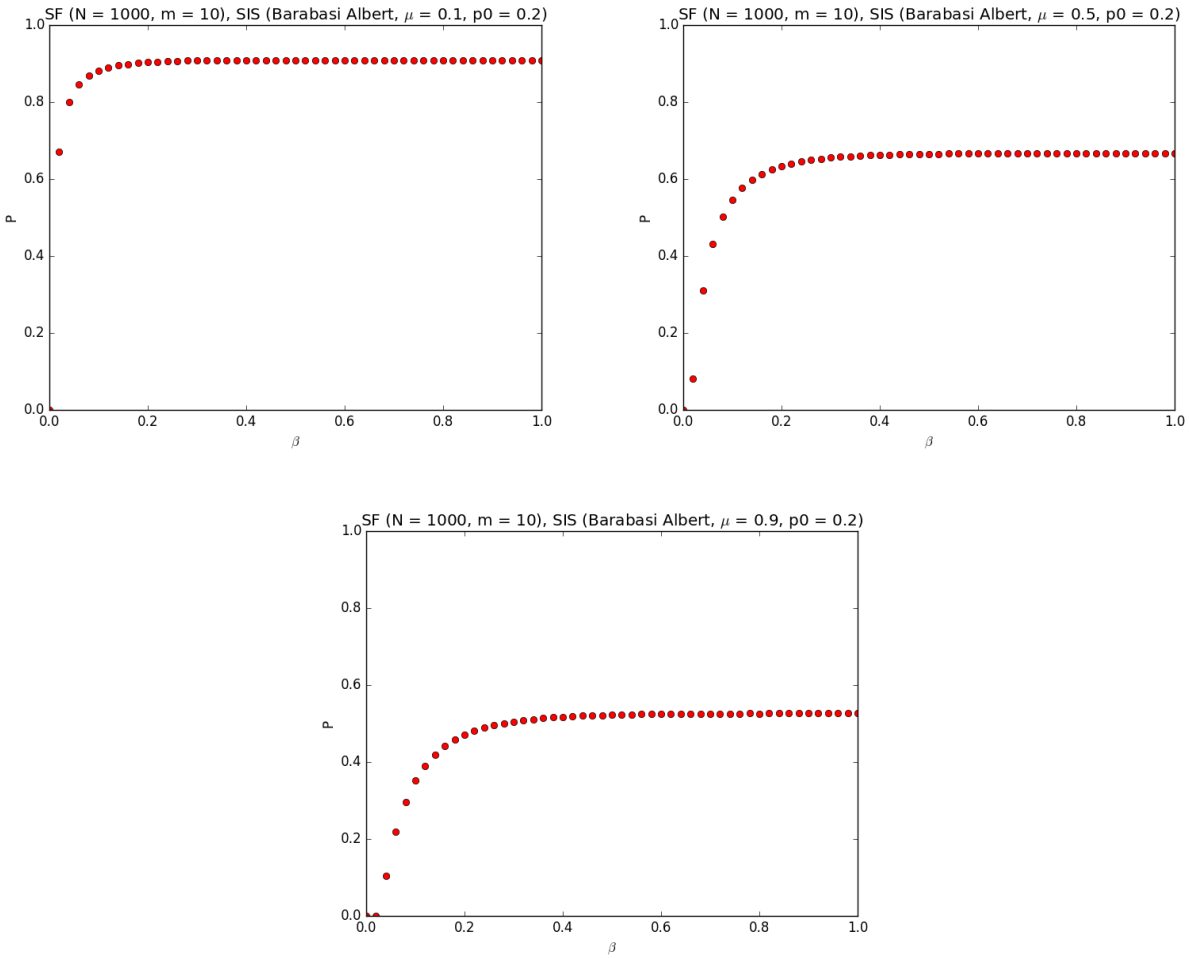
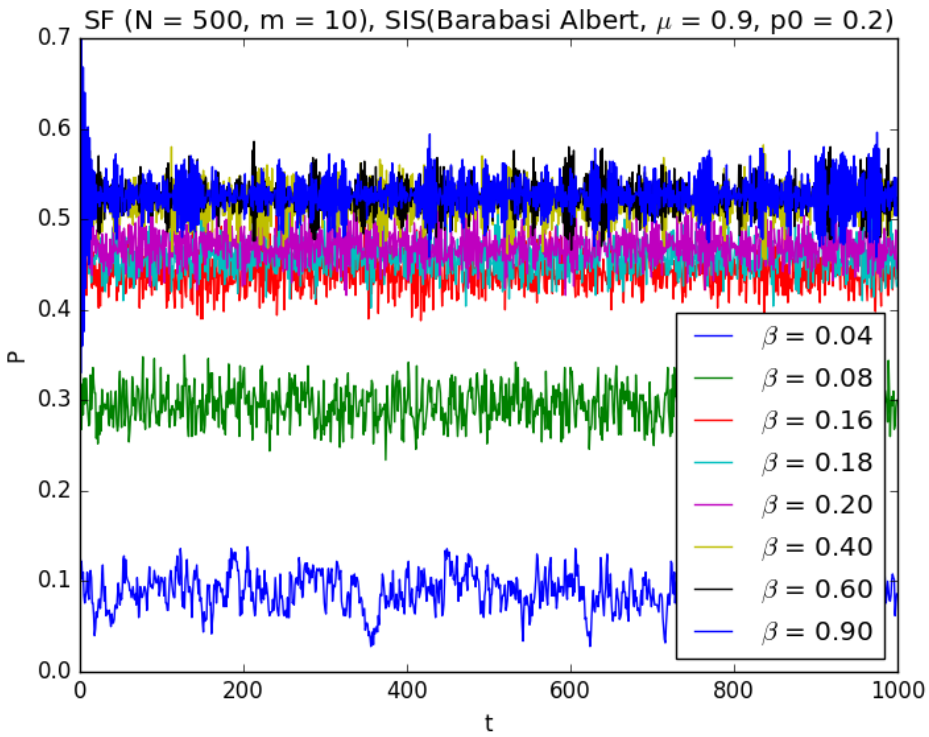
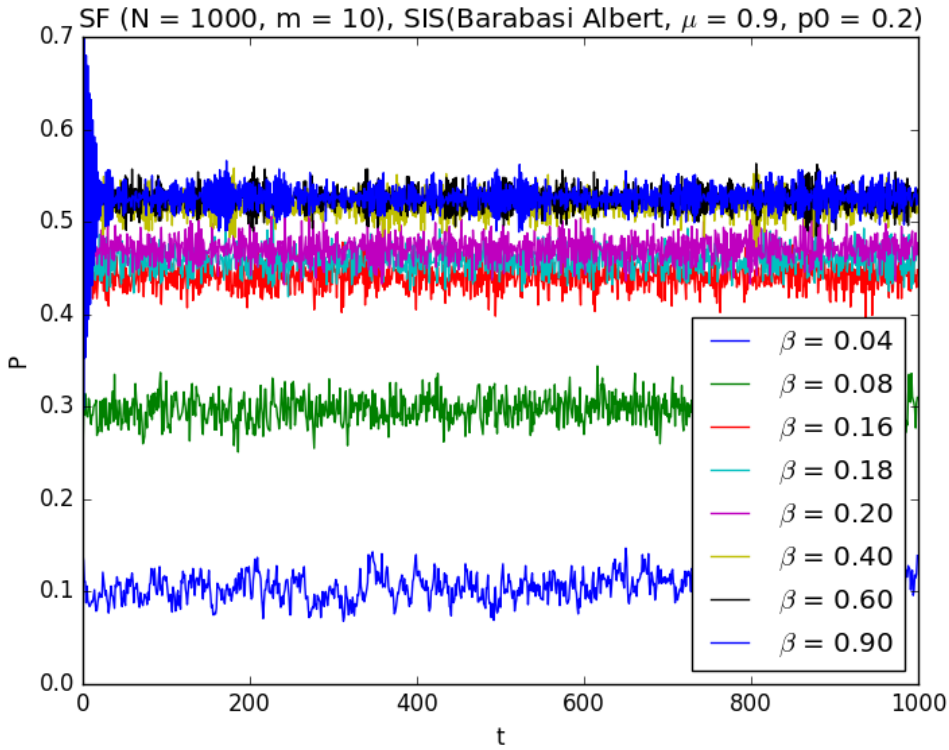


Figure 4: Plots of Barabasi Albert 1000 edges. P as a function of Beta.

Simulation plot



Simulation plot: Barabasi Albert 500 edges



Simulation plot: Barabasi Alber 1000 edges.

Random network

500 edges

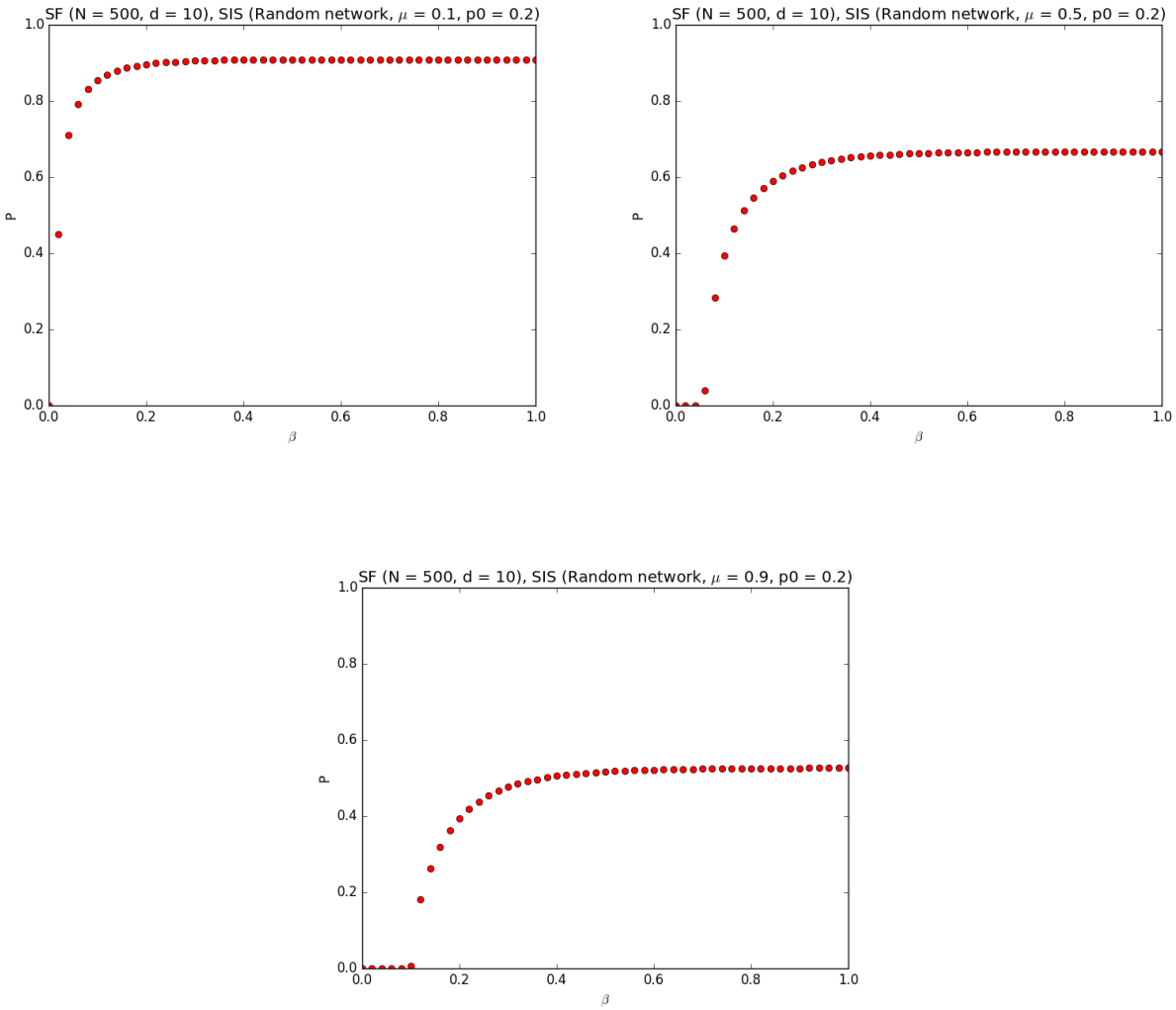


Figure 5: Plots of Random network 500 edges. P as a function of Beta.

1000 edges

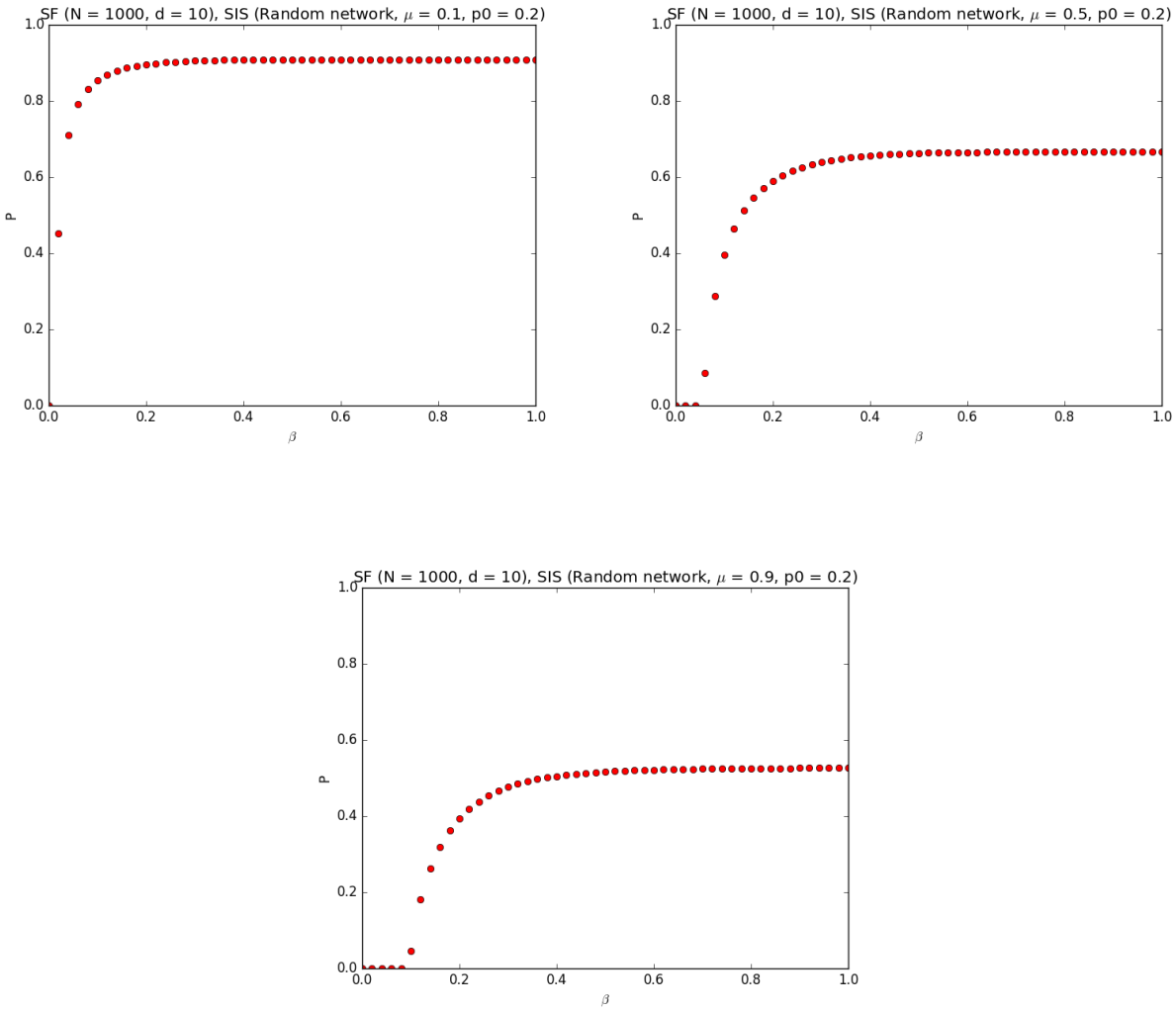
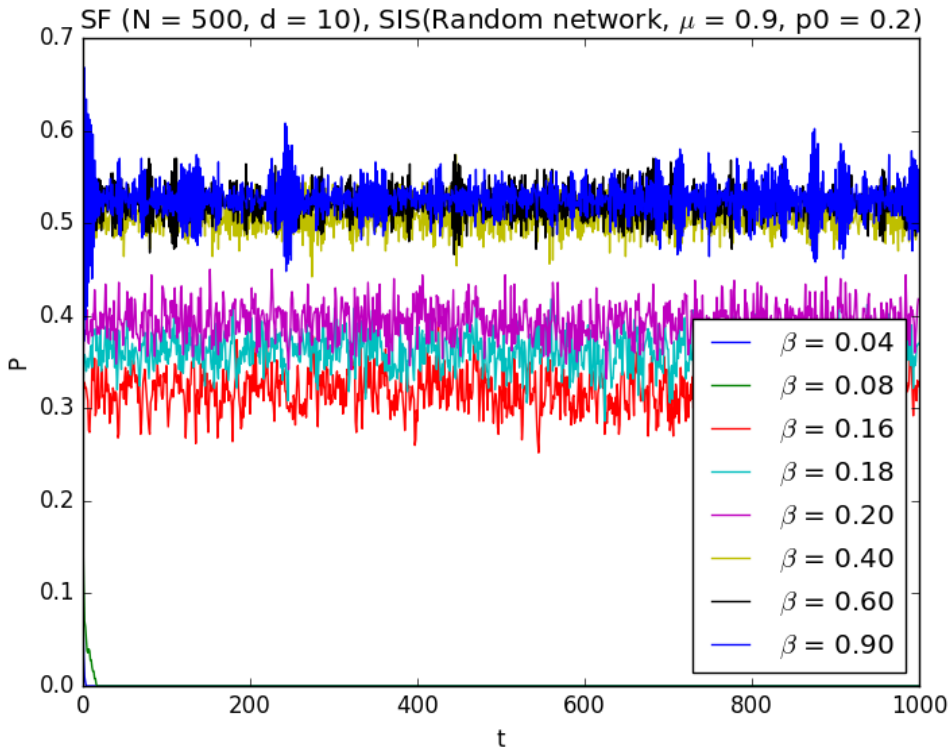
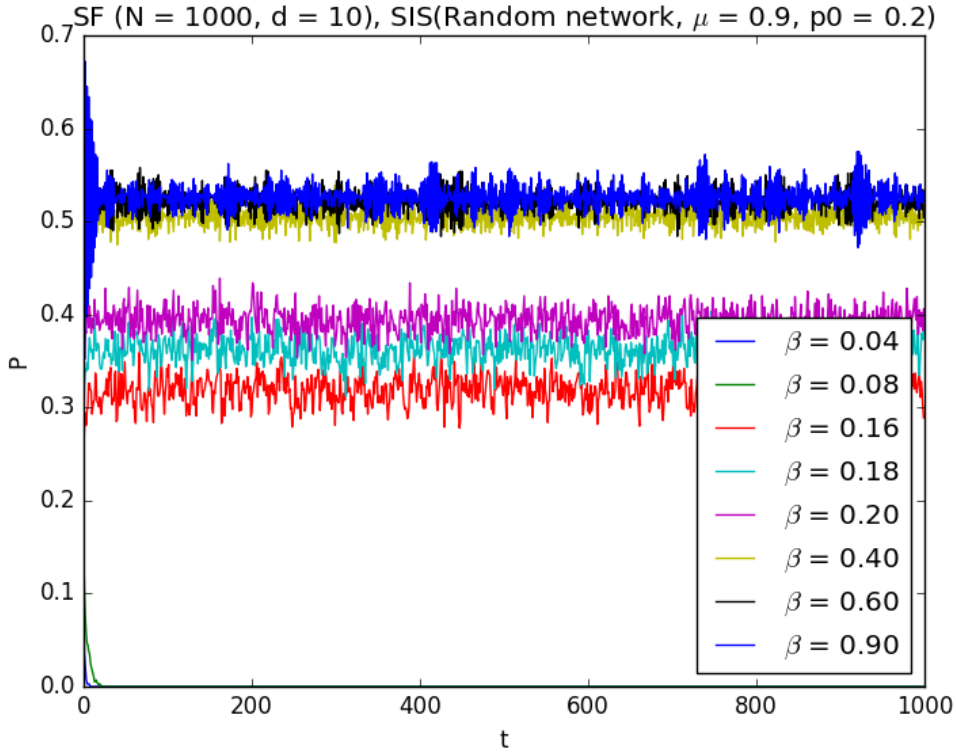


Figure 6: Plots of Random network 1000 edges. P as a function of Beta.

Simulation plot



Simulation plot: Random network 500 edges



Simulation plot: Random network 1000 edges.