Home work 3 - SDS

An SDS $\boldsymbol{S}$ is defined as a tuple ($\boldsymbol{G}$, $\boldsymbol{F}$, $\boldsymbol{\pi}$), where: (a) $\boldsymbol{G}$=(V, E) is the underlying graph on a set V of nodes, $|V| = n$ and $|E| = m$, (b) $\boldsymbol{F} = (f_v)$ denotes a set of local functions for each node v in V, on some fixed domain. Each node v computes its state by applying the function f_v on the states of its neighbors. (c) $\boldsymbol{\pi}$ denotes a permutation on V, and specifies the order in which the node states are to be updated by applying the appropriate function in $\boldsymbol{F}$. One update of the SDS involves applying the local functions in the order specified by $\boldsymbol{\pi}$. An SyDS is a variant of SDS in which nodes are updated simultaneously (synchronously). Here we will use **D** to denote the domain of the local functions in $\boldsymbol{F}$ and we will assume that all functions are over a Boolean domain {0, 1}. Thus each function f_v takes as input the value of v and its neighbors (all from the set {0, 1}) and returns either a 0 or a 1. A configuration **C** of an SDS (SyDS) $\boldsymbol{S}$ is an n-vector ($s_1, s_2, \ldots, s_n$), where each s_i is from **D** and denotes the state of node i . A single update of an SyDS (SDS) can be thought of as a function $\mathbf{H}: \{0,1\}^n \rightarrow \{0,1\}^n$ that takes a configuration C_1 and returns a new configuration C_2 by applying each of the node functions synchronously (or in the order given by the specified permutation). A phase space PS of an SDS (SyDS) $\boldsymbol{S}$ is a directed graph $G_1 = (V_1, E_1)$, in which each node is a configuration X of $\boldsymbol{S}$ and there is a directed edge from a configuration X to a configuration Y if $\mathbf{H}(X)$ yields Y, i.e., the dynamical system starting at X reaches Y in one step.

A predecessor of a configuration X is a configuration Y such that $\mathbf{H}(Y)$ returns X. A fixed point is a configuration X such that $\mathbf{H}(X)$ returns X. A Garden of Eden configuration is one that has no predecessors.

A threshold-SDS (SyDS) is one in which all of the local functions are simple Boolean threshold functions (recall that a t-threshold function is a Boolean function that returns a 0 when less than t of its inputs are 1 and returns 1 otherwise).

Question 1: Given an SDS (SyDS) $\boldsymbol{S}$ with n nodes, how many nodes does the phase space graph G_1 have?

Question 2: Consider a simple SDS (SyDS) with 5 nodes arranged in a simple cycle, i.e., the nodes of the SDS (SyDS) are $\{v_1, v_2, v_3, v_4, v_5\}$, and the edges are $\{(v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_5), (v_5, v_1)\}$. When viewing this as an SDS let $\boldsymbol{\pi}$ be $[v_1, v_2, v_3, v_4, v_5]$. Let each node compute a 2-threshold function.

1. Draw the phase space of the SDS and SyDS.
2. Count the number of fixed points and Garden of Eden configurations for SDS and the SyDS.
3. What is the length of the longest path in the phase space of the SDS and SyDS.
4. What is the predecessor of the configuration (0,0,0,0,0) and (1,1,1,1,1)?
5. Suppose I change my permutation from $[v_1, v_2, v_3, v_4, v_5]$ to $[v_2, v_1, v_4, v_3, v_5]$. Draw the phase space of this new SDS.

6. Repeat the above computations with modified SDS (SyDS) in which we add additional edges (v_1, v_3) and (v_2, v_4).

Question 3: Design an SyDS with four nodes or less that has the property that has a cycle of length 2 in its phase space. Do you think you can design a SDS with a similar property?

Question 4: Write a program that takes as input a number n (the number of nodes) and constructs *a small world network* on them: you arrange the nodes in a simple cycle and then add an edge between two nodes with probability p that is proportional to $d(u, v)^{-r}$, i.e., for say r =1 edges are added between vertices inversely proportional to their distance in the graph. We will consider just 3 values or r: 1, 2, and 4. Define p such that in expectation each node should have approximately $n/4$ edges incident on it in addition to the 2 edges that make up its neighbors. Think how would you do this (hint: you have 2 vertices at distance 2, 2 at distance 3, etc.). Each edge between vertices at distance i has the same probability of being present $1/i$ (in case r = 1). Then appropriate sum of these probabilities should be $n/4$. Each node then has an $n/8$ threshold function, i.e., roughly it becomes 1 when ½ or more of its neighbors are 1.

1. Device an elementary algorithm to count the number of fixed points and Garden of Eden configuration. What is the largest n for which you can count the number of fixed points and Garden of Eden configurations?
2. Do you notice a sharp increase in the running time as a function of n (plot it for n = 6, 7, 8, 9, 10, …) Can you think why?