Project 3 Tutorial

Cellular Automata

The dynamic behavior of a cellular automaton can be described by a truth table of the form

$x_{i-1}(k)$	$x_i(k)$	$x_{i+1}(k)$	$x_i(k+1)$
0	0	0	α_0
0	0	1	α_1
0	1	0	α_2
0	1	1	α_3
1	0	0	α_4
1	0	1	α_5
1	1	0	α_6
1	1	1	α_7

Table 1. Generic description of a cellular automaton.

where the last column defines the “rule”. Given that there are $2^8 = 256$ possible combinations of coefficients $\alpha_0 - \alpha_7$, it follows that there are exactly 256 different types of one-dimensional cellular automata. Each of them is normally indexed by a single number between 0 and 255, which is computed as:

$$N = \alpha_7 \cdot 2^7 + \alpha_6 \cdot 2^6 + \alpha_5 \cdot 2^5 + \alpha_4 \cdot 2^4 + \alpha_3 \cdot 2^3 + \alpha_2 \cdot 2^2 + \alpha_1 \cdot 2^1 + \alpha_0 \cdot 2^0$$

We can also describe cellular automata by representing the corresponding rule as a vector

$$R = \left[\begin{array}{cccccccc} \alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 & \alpha_7 \end{array} \right]$$

This approach will prove to be convenient for Matlab simulations.

If the automaton consists of L cells, the overall system will have 2^L possible states, each of which corresponds to a different combination of zeros and ones. Given any such combination

$$X(k) = \left[\begin{array}{cccc} x_0(k) & x_1(k) & \ldots & x_{L-1}(k) \end{array} \right]$$

we can use the truth table to compute the next state of the system

$$X(k+1) = \left[\begin{array}{cccc} x_0(k+1) & x_1(k+1) & \ldots & x_{L-1}(k+1) \end{array} \right]$$

Note. Since cells 0 and $L-1$ have only two neighbors in the row above, we have to assume that there is a “wrap around” connection between these two cells (otherwise, we wouldn’t be able to apply the truth table uniformly). This effectively means that $x_0(k+1)$ is influenced by $x_{L-1}(k)$, $x_0(k)$ and $x_1(k)$, and that $x_{L-1}(k+1)$ depends on $x_{L-2}(k)$, $x_{L-1}(k)$ and $x_0(k)$.

A simple way to describe the overall behavior of the automaton is to form a matrix M whose first column consists all posible states $X(k)$ (in decimal form), and whose second column represents the states $X(k+1)$ that succeed them. The function M = matrixM(R,L) is designed to produce such a matrix for a given rule R and automaton length L. The following example illustrates how we can use this matrix to schematically represent all possible limit cycles and their basins.

Example 1. Suppose that $L = 4$, and that function M = matrixM(R,L) produces matrix

$$M = \begin{bmatrix} 0 & 0 \\ 1 & 3 \\ 2 & 6 \\ 3 & 6 \\ 4 & 12 \\ 5 & 15 \\ 6 & 14 \\ 7 & 12 \\ 8 & 9 \\ 9 & 3 \\ 10 & 15 \\ 11 & 6 \\ 12 & 13 \\ 13 & 7 \\ 14 & 11 \\ 15 & 0 \end{bmatrix}$$

This matrix allows us to start from any of the 16 possible states, and follow the evolution of the system until it reaches a limit cycle. If we start from state $X(0) = 5$, for example, we know that $X(1) = 15$, since $M(5,2) = 15$. Similarly, from the fact that $M(15,2) = 0$ it follows that $X(2) = 0$. The first row of matrix M indicates that the system remains in this state in all subsequent steps, so we can conclude that state 5 belongs to the basin of limit cycle $\{0\}$.

It is usually helpful to aggregate the information contained in matrix M into a schematic diagram such as the one shown in Fig. 1. From this figure, we can deduce that the system has three attractive limit cycles:

Limit Cycle 1: $\{0\}$, with basin $\{5, 10, 15\}$

Limit Cycle 2: $\{6, 14, 11\}$, with basin $\{1, 2, 3, 8, 9\}$

Limit Cycle 3: $\{7, 12, 13\}$, with basin $\{4\}$

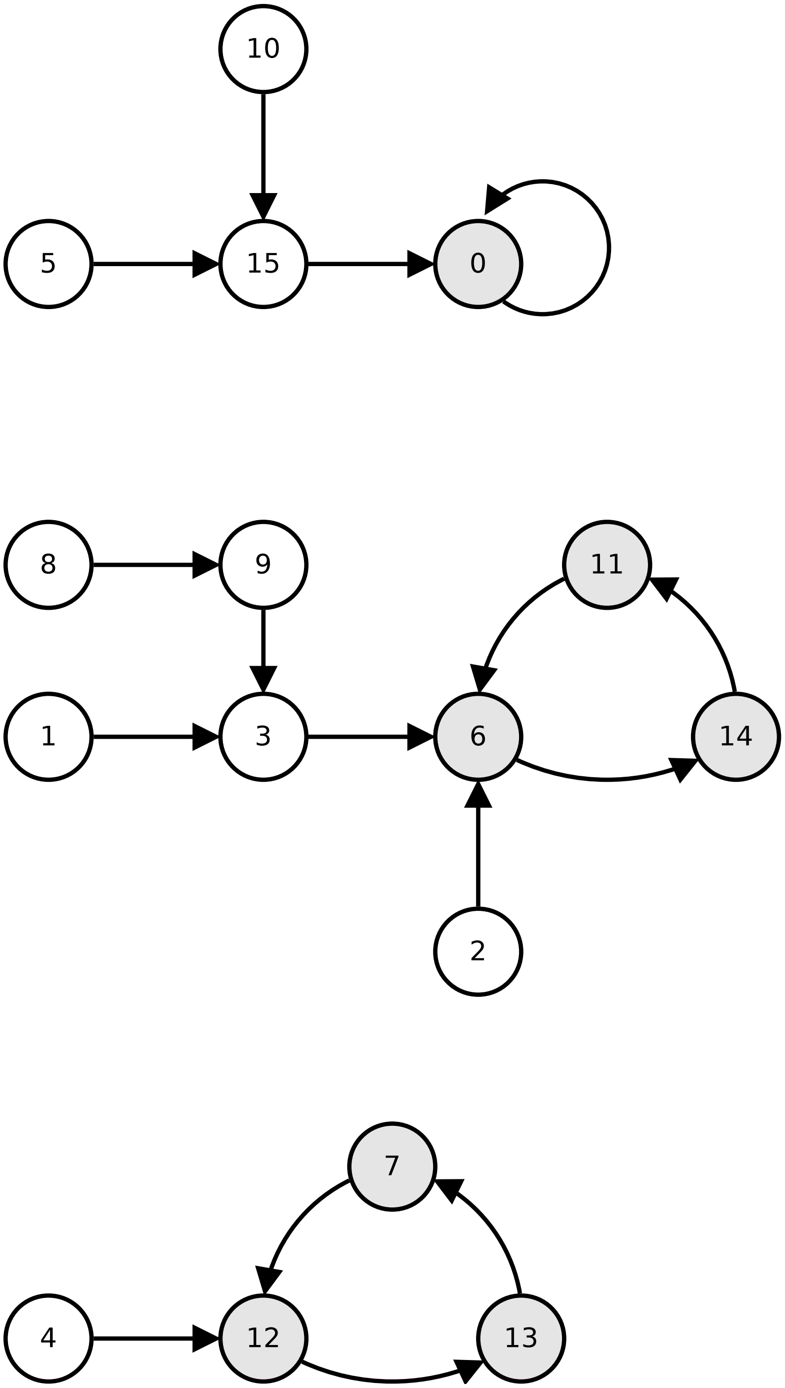

Figure 1: Limit cycles that correspond to matrix M.

Note. When working with large values of L, it is not very convenient to form matrix M directly, since its dimensions are $2^L \times 2$. Instead, it is better to use a software package such as the one available at http://www.cellularautomatagenerator.com/. In order to do that, it is necessary to specify the rule schematically, in the manner shown in Fig. 2 (this particular configuration corresponds to Rule 132, whose binary equivalent is 10000100).

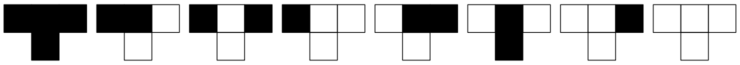

Figure 2: Schematic representation of Rule 132.

Boolean Networks

Consider the simple Boolean network shown in Fig. 3, which has $N = 3$ and $K = 2$. We will assume that the associated functions $F_i(x(k))$ are the ones indicated in Tables 2-4. Using these three definitions, we can easily describe how the overall state vector $X(k) = [\, x_1(k)\ x_2(k)\ x_3(k)]$ evolves over time (this is shown in Table 5).

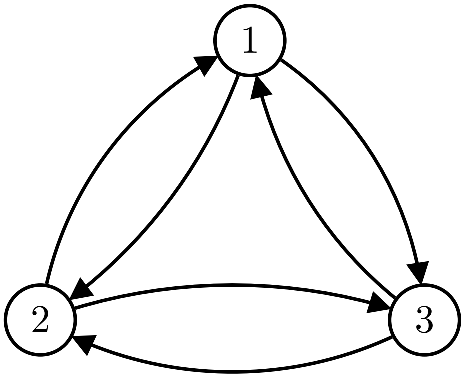

Figure 3: Boolean network with $N = 3$ and $K = 2$.

$x_2(k)$	$x_3(k)$	$F_1\,[x_2(k), x_3(k)]$
0	0	0
0	1	0
1	0	0
1	1	1

Table 2. Function $F_1\,[x(k)]$.

$x_1(k)$	$x_3(k)$	$F_2\left[x_1(k), x_3(k)\right]$
0	0	0
0	1	0
1	0	0
1	1	1

Table 3. Function $F_2\left[x(k)\right]$.

$x_1(k)$	$x_2(k)$	$F_3\left[x_1(k), x_2(k)\right]$
0	0	0
0	1	1
1	0	1
1	1	1

Table 4. Function $F_3\left[x(k)\right]$.

$x_1(k)$	$x_2(k)$	$x_3(k)$	$x_1(k+1)$	$x_2(k+1)$	$x_3(k+1)$
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	0	0	1
1	0	1	0	1	1
1	1	0	0	0	1
1	1	1	1	1	1

Table 5. Evolution of state $X(k)$.

As in the case of cellular automata, the system dynamics can be described in compact form using matrix M, whose columns represent overall states $X(k)$ and $X(k+1)$, respectively. It is not difficult to see that Table 5 corresponds to matrix

$$M = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 2 & 1 \\ 3 & 5 \\ 4 & 1 \\ 5 & 3 \\ 6 & 1 \\ 7 & 7 \end{bmatrix}$$

We can use this matrix to obtain a schematic description of the possible limit cycles and their basins (such a description is shown in Fig. 4). Note that only one of these cycles is attractive - the other two are Isles of Eden.

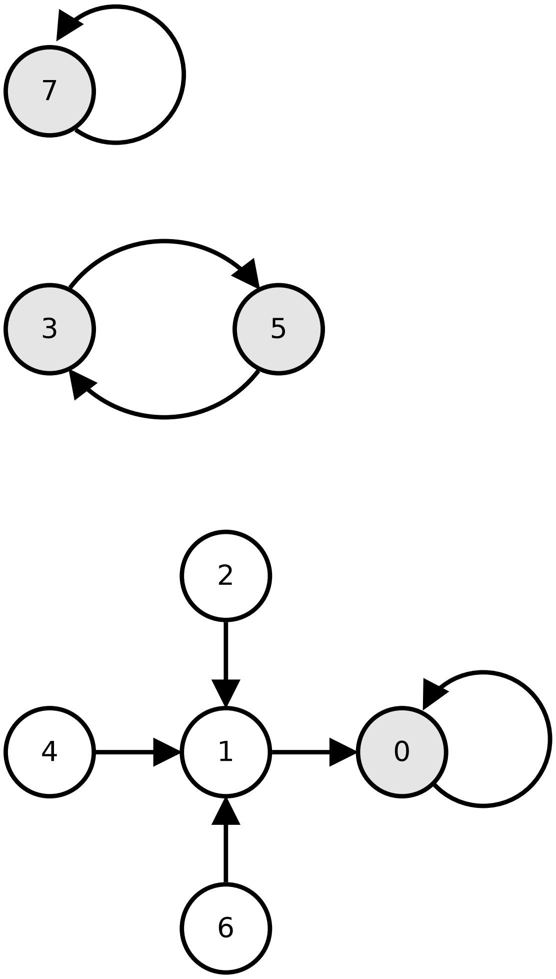

Figure 4: Limit cycles that correspond to matrix M.

For a given N and K, matrix M can be obtained using function

$$\mathrm{M} = \mathrm{BooleanM(R, U, K, N)}$$

where row i of matrix R corresponds to function F_i, and row i of matrix U represents the nodes that influence node i. In this compact format, the Boolean functions in Tables 2-4 can be represented as

$$R = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

and the connectivity of the network in Fig. 3 can be described using matrix

$$U = \begin{bmatrix} 2 & 3 \\ 1 & 3 \\ 1 & 2 \end{bmatrix}$$

For larger values of N, it makes little sense to form matrix M, since the number of possible states grows as 2^N. In such cases, the best we can do is perform a statistical analysis of the system behavior, using a set of randomly chosen matrices R and U. For each such pair, we can estimate the number and length of attractive limit cycles by simulating the system starting from Q different initial states. We may not be able to indentify all the cycles in this way, but we can get pretty close when Q is large.

Once we have examined a sufficient number of configurations, we can average the obtained results, and draw some general conclusions about how NK networks behave. Function

$$[\text{Avg\_NUM, Avg\_L}] = \text{BoolStats(N, K, Q)}$$

is designed to perform this type of analysis (in order to reduce the computation, we usually set $Q = 150$). In principle, this function can produce the average number and average length of attractive limit cycles for any choice of N and K. However, the execution time becomes excessively large for $N > 50$, so we will consider only cases where $N \leq 50$ and $K \leq 3$.