ENS'07 *Paris, France, 3-4 December 2007*

A DYNAMIC NETWORK MODEL FOR NANOSCALE SYSTEMS

Elena Dubrova   *Hannu Tenhunen*

Royal Institute of Technology, IMIT/KTH, Stockholm, Sweden
{elena, hannu}@imit.kth.se

ABSTRACT

Much current research is concentrated on building computing networks based on nanoscale devices. However, there has been less progress in programming these networks to produce useful computation. This paper considers a model of computation based on Random Boolean Networks (RBNs). RBNs have the ability to form complex dynamic patterns in ways that produce collective information processing. This approach to information processing is very different from classical approaches, and requires non-standard synthesis and analysis techniques. A random network-based model seems to be appealing for emerging nanotechnologies in which it is difficult to control the growth direction or achieve precise assembly. We formally define the model, discuss its universality, and investigate an evolutionary method for "programming" RBNs to perform computations.

1. INTRODUCTION

Random Boolean Networks (RBNs) were introduced by Kauffman in 1969 in the context of gene expression and fitness landscapes. Each vertex in an RBN represents a gene. An edge from one vertex to another implies a causal link between the two genes. "On" state of a vertex corresponds to the gene being expressed. Time is viewed as proceeding in discrete steps. At each step, the new state of a vertex is a Boolean function of the previous states of its predecessors.

Kauffman has shown that it is possible to tune the parameters of an RBN so that it exhibits self-organized critical behavior ensuring both stability and evolutionary improvements. Statistical features of self-organized RBNs match the characteristics of living cells. The number of cycles in the network's state space, called attractors, corresponds to the number of different cell types. The attractor's length corresponds to the cell cycle time. The sensitivity of attractors to different kind of disturbances, modeled by changing a network connection, a state of a vertex, or the associated function, reflects the ability of the cell to resist damage, mutations and virus attacks.

RBNs were also applied to the problems of cell differentiation [1], immune response [2], evolution [3], and neural networks [4, 5]. They have attracted the interest of physicists due to their analogy with the disordered systems studied in statistical mechanics, such as the mean field spin glass [6, 7, 8].

In this paper, we investigate how RBN can be used for computing logic functions. We formally define the model, discuss its universality, and investigate an evolutionary method for "programming" RBNs to compute a given function.

Our motivation for considering random networks is that they seem to be an appealing mathematical model for emerging nano-scale technologies in which it is difficult to control the growth direction or achieve precise assembly, e.g. carbon nanotubes. It has been demonstrated that random arrays of carbon nanotubes are much easier to produce compared to the ones with a fixed structure [9]. Random arrays of carbon nanotubes can be deposited at room temperature onto polymeric and many other substrates, which makes them a promising new material for lightweight flexible displays, smart materials or clothing, biological and chemical sensors, tunable frequency-selective surfaces, etc. Conventional semiconductors are not suitable for such applications because they and too expensive and require a crystalline substrate. The effort to developing organic semiconductors has achieved only moderate success so far, mostly because of the low-quality electron transport of organic semiconductors. Random arrays of carbon nanotubes provide a high-quality electron transport and therefore can be a much better alternative.

The paper is organized as follows. Section 2 gives a definition of RBNs and summarizes their properties. Section 3 describes how we can use RBNs for computing logic functions. Section 4 addresses the problem of constructing an RBN for a given function. Section 5 concludes the paper and discusses open problems.

2. BACKGROUND AND PREVIOUS WORK

In this section, we give a brief introduction to Random Boolean Networks. For a more detailed description, the reader is referred to [5].

2.1. Definition of RBN

An (n, k)-*Random Boolean Network* is a synchronous Boolean automaton with n vertices. Each vertex v has k predecessors, assigned independently and uniformly at random from the set of all vertices, and an associated Boolean function $f_v : \{0,1\}^n \rightarrow \{0,1\}$. Functions are selected so that they evaluate to values 0 and 1 with given probabilities p and $1 - p$, respectively. Time is viewed as proceeding in discrete steps. At each step, the next value of the state variable x_v associated with a vertex v is a function of the previous values of the state variables x_{u_i} associated with the predecessors of v, u_i, $i \in \{1, 2, \ldots, k\}$:

$$x_v^+ = f_v(x_{u_1}, x_{u_2}, \ldots, x_{u_k}),$$

A state of an RBN is defined by the ordered set of values of the state variables associated with its vertices.

An example of an RBN with $n = 10$ and $k = 2$ is shown in Figure 1. We use "$\cdot$", "$+$" and "$'$" to denote the Boolean operations AND, OR and NOT, respectively.

2.2. Frozen and chaotic phases

The parameters k and p determine the dynamics of an RBN. If a vertex controls many other vertices, and the number of controlled vertices grows in time, the RBN is said to be in a *chaotic phase* [10]. Typically such a behavior occurs for large values of $k \sim n$. The next states of the RBN are random with respect to the previous ones. The dynamics of the network is very sensitive to changes in the values of state variables, associated Boolean function, or network connections.

If a vertex controls only a small number of other vertices and their number remains constant in time, the RBN is said to be in a *frozen phase* [11]. Usually, independently on the initial state, after a few steps, the network reaches a stable state. This behavior usually occurs for small values of k, such as $k = 0$ or 1.

There is a critical line between the frozen and the chaotic phases, when the number of vertices controlled by a vertex grows in time, but only up to a certain limit [12]. Statistical features of RBNs on the critical line are shown to match the characteristics of real cells and organisms [13, 14]. The minimal disturbances typically create only small variations in the network's dynamics. Just some rare perturbations evoke radical changes.

For a given probability p, there is a critical number of inputs k_c below which the network is in the frozen phase and above which the network is in the chaotic phase [6]:

$$k_c = \frac{1}{2p(1-p)}. \qquad (1)$$

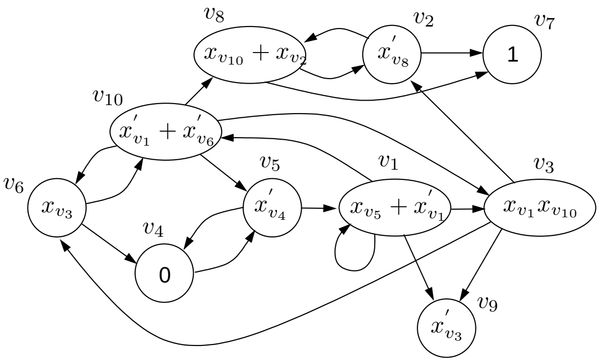

Fig. 1. An example of an $(10, 2)$-RBN.

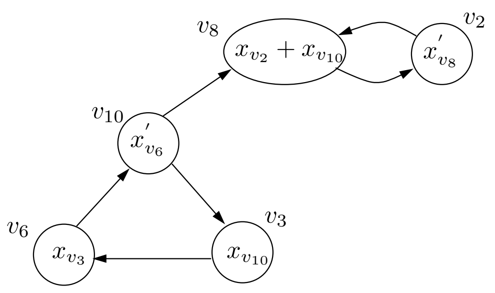

Fig. 2. The reduced network for the RBN in Figure 1.

2.3. Attractors

An infinite sequence of consecutive states of a network is called a *trajectory*. A trajectory is uniquely defined by the initial state. Since the number of possible states is finite, all trajectories eventually converge to either a single state, or a cycle of states, called *attractor*. The *basin of attraction* of A, denoted by $B(A)$, is the set of all trajectories leading to the attractor A.

A number of algorithms for finding attractors in RBNs have been presented. Most of them are based on an explicit representation of the set of states on an RBN and therefore are applicable to networks with up to 32 vertices only [15, 16, 17, 18]. The algorithm presented in [19] uses an implicit representation, namely Binary Decision Diagrams [20], and can handle larger RBNs. It is a very efficient algorithm which finds the set of states of all attractors simultaneously without computing the rest of the states.

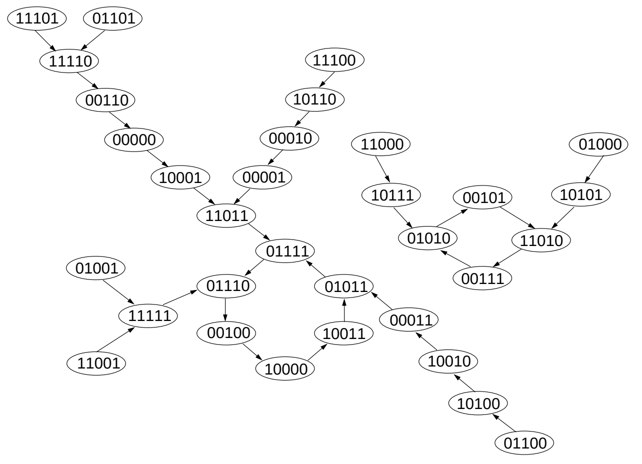
Fig. 3. The state transition graph of the RBN in Figure 2. The states are ordered as $(x_{v_2}x_{v_3}x_{v_6}x_{v_8}x_{v_{10}})$.

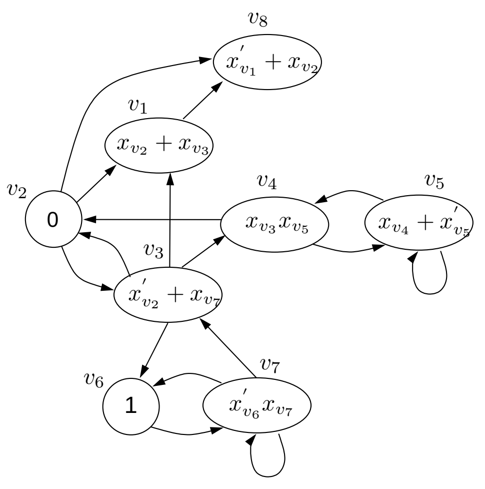
Fig. 4. An $(8, 2)$-RBN for the 2-input AND.

2.4. Redundant Vertices

It is possible to reduce the state space of an RBN by removing redundant vertices which have no influence on network's dynamics. A vertex v is considered *redundant* for an RBN G if the network G_R obtained by removing v form G has the same number and length of attractors as G. If a vertex is not redundant, it is called *relevant*.

There are several types of redundant vertices. First, all vertices v whose associated function f_v is constant 0 or constant 1 are redundant. If u is an successor of an redundant vertex v and if after the substitution of the constant value of f_v in f_u the function f_u reduces to a constant, then u is redundant, too.

Second, all vertices v which have no successors are redundant. If u is a predecessor of an redundant vertex v and if all successors of u are redundant, then u is redundant, too.

Third, a vertex can be redundant because its associated function f_v has a constant value due to the correlation of its input variables. For example, if a vertex v with an associated OR (AND) function has predecessors u_1 and u_2 with functions $f_{u_1} = x_w$ and $f_{u_2} = x'_w$, then the value of f_v is always 1 (0). This kind of redundant vertices are hardest to identify.

Exact and approximate bounds on the size of the set of relevant vertices for different values of k and p have been given [21, 11, 12, 10, 22]. In the infinite size limit $n \to \infty$, in the frozen phase, the number of relevant vertices remains finite. In the chaotic phase, the number of relevant vertices is proportional to n. On the critical line, the number of relevant vertices scales as $n^{1/3}$ [15].

The algorithms for computing the set of all redundant vertices, e.g. [16], are too computationally expensive and therefore are feasible for RBNs with up to a thousand vertices only. The *decimation procedure* presented in [18] computes only a subset of redundant vertices, but it is applicable to large networks. In time linear in the size of an RBN it finds redundant vertices evident from the structure of the network (1st and 2nd type). The decimation procedure will not identify the redundant vertices whose associated functions have constant values due to the correlation of their input variables (3rd type).

The reduced network for the RBN in Figure 1 is shown in Figure 2. Its state transition graph is given in Figure 3. Each vertex of the state transition graph represents a 5-tuple $(x_{v_2}x_{v_3}x_{v_6}x_{v_8}x_{v_{10}})$ of values of states on the relevant vertices v_2, v_3, v_6, v_8, v_{10}. There are two attractors: $\{01111, 01110, 00100, 10000, 10011, 01011\}$ and $\{00101, 11010, 00111, 01010\}$.

3. COMPUTING LOGIC FUNCTIONS WITH RBNS

In this section we discuss how RBNs can be used for computing logic functions. One possibility is to use state variables of RBN vertices to represent function's variables, and to map attractors into the function's values.

Suppose that we have an (n, k)-RBN which has m attractors $A_0, A_1, \dots, A_{m-1}$. The basins of attractions $B(A_0)$, $B(A_1), \dots, B(A_{m-1})$ partition the Boolean space $\{0,1\}^n$ into m connected components via a dynamic process. Attractors constitute stable equilibrium points. If we assign a value i, $i \in \{0, 1, \dots, m-1\}$ to the attractor A_i and assume that the set of minterms represented by the states in the basin of attraction of A_i is mapped to i, then, the RBN represents the function $f : \{0,1\}^r \to \{0, 1, \dots, m-1\}$ of variables $x_1, \dots, x_n$, where the variable x_i corresponds to the state variable of the vertex v_i. If $m = 2$, then the RBN represents a Boolean function.

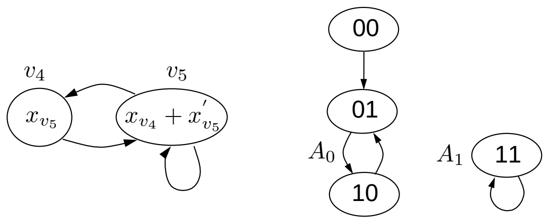

Fig. 5. The reduced network for the RBN in Figure 4 and its state transition graph. The state are ordered as $(x_{v_4} x_{v_5})$.

Definition 1 *An (n,k)-RBN with m attractors represents the function of type $f : \{0,1\}^n \rightarrow \{0,1,\dots,m-1\}$ which is defined as follows:*

$$(a_1,\dots,a_n) \in B(A_i) \quad \Rightarrow \quad f(a_1,\dots,a_n) = i,$$

$\forall (a_1,\dots,a_n) \in \{0,1\}^n, \forall i \in \{0,1,\dots,m-1\}$,

Note, some variables of f may be redundant. The number of vertices on which f actually depends is equal to the number of relevant vertices of the RBN.

As an example, consider the (8,2)-RBN shown in Figure 4. The vertices v_4 and v_5 are relevant vertices, determining the dynamic of the RBN according to the reduced network in Figure 5(a). The state transition graph of the reduced network is shown in Figure 5(b). There are two attractors, A_0 and A_1. We assign the logic 0 to A_0 and the logic 1 to A_1. The initial states $00, 01$ and 10 terminate in the attractor A_0 (logic 0) and the initial state 11 terminates in the attractor A_1 (logic 1). So, the RBN represents the 2-input AND.

As another example, consider the RBN in Figure 2 and its state transition graph in Figure 3. If we assign the logic 0 to the left-hand side attractor and the logic 1 to the right-hand side one, then we get the Boolean function

$$f = x_2 x'_3 x'_5 + x'_2 x_3 x_4 (x_1 + x_5).$$

The RBN representation described by Definition 1 is not unique since we can find many different RBNs representing the same function. For example, the reduced network in Figure 6 has the same state transition graph as the one in Figure 5.

One can easily show that an RBN representation with two attractors exists for any n-variable Boolean function, since we can always construct a trivial (n,n)-RBN as follows. Choose any assignment $(a_1,\dots,a_n) \in \{0,1\}^n$ of variables of f such that $f(a_1,\dots,a_n) = 0$. Assign $(a_1,\dots,a_n)$ to be the next state of every state $(b_1,\dots,b_n)$ of RBN for which $f(b_1,\dots,b_n) = 0$.

Similarly, choose any assignment $(c_1,\dots,c_n) \in \{0,1\}^n$ of variables of f such that $f(c_1,\dots,c_n) = 1$. Assign $(c_1,\dots,c_n)$ to be the next state of every state $(d_1,\dots,d_n)$ of RBN for which $f(d_1,\dots,d_n) = 0$.

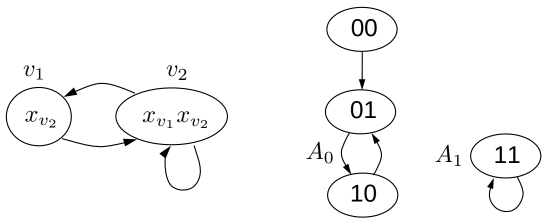

Fig. 6. An alternative network for the 2-input AND and its state transition graph. The states are ordered as $(x_{v_1} x_{v_2})$.

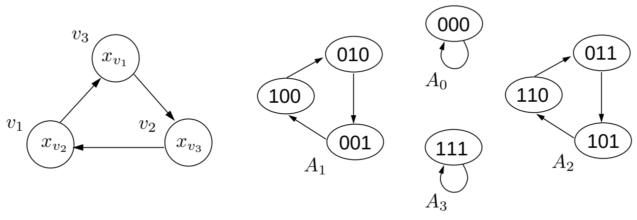

Fig. 7. An (3,1)-RBN for the 3-variable majority function and its state transition graph. The states are ordered as $(x_{v_1} x_{v_2} x_{v_3})$.

By construction, the resulting RBN has two single-vertex attractors: $A_0 = (a_1,\dots,a_n)$ and $A_1 = (c_1,\dots,c_n)$ and the following associated functions $f_{v_1},\dots,f_{v_n}$:

- $f_{v_i} = f$ if $a_i = 0$ and $c_i = 1$;
- $f_{v_i} = f'$ if $a_i = 1$ and $c_i = 0$;
- $f_{v_i} = 0$ if $a_i = 0$ and $c_i = 0$;
- $f_{v_i} = 1$ if $a_i = 1$ and $c_i = 1$.

It is desirable to minimize the input degree k of an RBN as much as possible, ideally to $k = 2$, so that a complex functionality is obtained from simpler primitives. However, some functions require $k = n$ for an RBN with two attractors to exist. One example of such function is a 3-variable majority function. Any (n,k)-RBN representing it with $k < 3$ has at least 4 attractors (see Figure 7 and Table 1 for an example).

It is possible, however, to find a $(n,3)$-RBN for the majority such that the functions associated to the vertices are simpler than the majority itself. Consider, for instance, the case shown in Table 2 and Figure 8. The vertex v_3 has a 3-input XOR associated to it. Other two vertices are 1- and 2-variable functions.

Rather than increasing the input degree of RBN vertices, we can work with multiple-valued output functions,

x_1	x_2	x_3	x_1^+	x_2^+	x_3^+	f
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	1	0
0	1	1	1	0	1	1
1	0	0	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	1	1	1
1	1	1	1	1	1	1

Table 1. A mapping of states for the 3-variable majority function resulting in 2 attractors and $k = 3$.

or, equivalently, map the same logic value to several attractors. Consider again the RBN representing the majority in Figure 7. We can assign four different values to the attractors and treat the resulting representation as a function of type $\{0,1\}^3 \rightarrow \{0,1,2,3\}$. Note that this function actually *counts* the number of 1's in the input assignment.

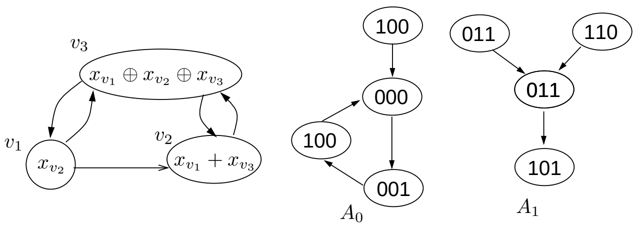

Fig. 8. An (3,3)-RBN for the 3-variable majority function and its state transition graph. The states are ordered as $(x_{v_1} x_{v_2} x_{v_3})$

x_1	x_2	x_3	x_1^+	x_2^+	x_3^+	f
0	0	0	0	0	1	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	1	0	1	1
1	0	0	0	0	0	0
1	0	1	1	0	1	1
1	1	0	0	1	1	1
1	1	1	1	1	0	1

Table 2. A mapping of RBN states for the 3-variable majority function resulting in 2 attractors and $k = 3$.

4. SYNTHESIS OF RBNS

In the traditional logic synthesis, functions are realized by composing them from simpler functions such as AND, OR,

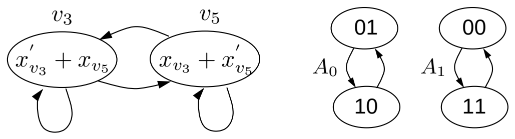

Fig. 9. The reduced network for the RBN in Figure 4 with three mutations (described in Section 4) and its state transition graph. The states are ordered as $(x_{v_3} x_{v_5})$.

NOT. Such a compositional approach does not apply easily to RBNs, because, as it was shown in [23], the number of attractors increases as a result of composition. If the RBN G_1 has p_1 attractors, the RBN G_2 has p_2 attractors, and G_1 and G_2 have no vertices in common, then then the RBN composed from G_1 and G_2 has $p1 \bullet p_2$ attractors, where $\bullet$ is the least common multiple of p_1 and p_2. So, by composing RBNs we increase the number of output values of the function represented by RBNs.

To avoid this problem, we use an evolutionary programming algorithm to construct an RBN for a given function directly. It starts from a population of randomly selected RBNs. In each generation, the fitness of every RBN in the population is evaluated by comparing (XORing) its function with the target function. Smaller number of 1's in the result implies a better fit. Then, multiple individuals are selected based on their fitness and modified by applying mutations to form a new population. The mutations are implemented by applying one of the following transformations, chosen at random, to a randomly selected vertex:

- a predecessor of a vertex v is changed, i.e. the edge (u, v) is replaced by an edge (w, v), $v, u, w \in V$;
- the Boolean function of a vertex is changed to a different Boolean function.

The selected new population starts another generation and the process is repeated.

As an illustration, suppose that the RBN representing the 2-input AND gate in Figure 4 is the starting point of the algorithm. If the the following three mutations are applied to this RBN:

1. edge (v_4, v_5) is replaced by (v_3, v_5);
2. edge (v_2, v_3) is replaced by (v_3, v_3);
3. edge (v_7, v_3) is replaced by (v_5, v_3).

then the resulting RBN represents the 2-input XNOR. The reduced version of this RBN is shown in Figure 9. WE assume that the logic 0 is assigned to A_1 and the logic 1 is assigned to A_2.

Our current version of the algorithms is too slow for large functions, because it re-computes the complete state space in order to evaluate the fitness function. We are searching for a possibility to make re-computations local in order to speed up the algorithm.

5. CONCLUSION AND OPEN PROBLEMS

In this paper, we present a model of computation in which states of the relevant vertices of an RBN represent variables of the function, and attractors represent function's values. Such a model seems to be appealing for emerging nanotechnologies in which it is difficult to control the growth direction or achieve precise assembly.

Future work includes developing a more efficient technique for "programming" RBNs to a functionality which minimizes input degree of RBN nodes, and investigating the "real-world" requirements of dealing with unreliable nodes or connections, and noisy systems.

We also plan to compare RBNs to other types of networks, with a different connectivity. The dynamics of a random network strongly depends on the way of selecting connections. In RBNs, each vertex has an equal probability of being connected to other vertices. Alternatively, in cellular automata [24], each vertex is connected only to its immediate neighbors, and all connections are arranged in a regular lattice. Intermediate cases are possible, for example, in small-world networks [25] some connections are to distant vertices and some to neighboring vertices.

6. REFERENCES

[1] S. Huang and D. E. Ingber, "Shape-dependent control of cell growth, differentiation, and apoptosis: Switching between attractors in cell regulatory networks," *Experimental Cell Research*, vol. 261, pp. 91–103, 2000.

[2] S. A. Kauffman and E. D. Weinberger, "The nk model of rugged fitness landscapes and its application to maturation of the immune response," *Journal of Theoretical Biology*, vol. 141, pp. 211–245, 1989.

[3] S. Bornholdt and T. Rohlf, "Topological evolution of dynamical networks: Global criticality from local dynamics," *Physical Review Letters*, vol. 84, pp. 6114–6117, 2000.

[4] H. Atlan, F. Fogelman-Soulie, J. Salomon, and G. Weisbuch, "Random Boolean networks," *Cybernetics and System*, vol. 12, pp. 103–121, 2001.

[5] M. Aldana, S. Coopersmith, and L. P. Kadanoff, "Boolean dynamics with random couplings," http://arXiv.org/abs/adap-org/9305001.

[6] B. Derrida and Y. Pomeau, "Random networks of automata: a simple annealed approximation," *Biophys. Lett.*, vol. 1, pp. 45, 1986.

[7] B. Derrida and H. Flyvbjerg, "Multivalley structure in Kauffman's model: Analogy with spin glass," *J. Phys. A: Math. Gen.*, vol. 19, pp. L1103, 1986.

[8] B. Derrida and H. Flyvbjerg, "Distribution of local magnetizations in random networks of automata," *J. Phys. A: Math. Gen.*, vol. 20, pp. L1107, 1987.

[9] E. S. Snow, J. P. Novak, P. M. Campbell, and D. Park, "Random networks of carbon nanotubes as an electronic material," *Applied Physics Letters*, vol. 81, no. 13, pp. 2145–2147, 2003.

[10] B. Luque and R. V. Sole, "Stable core and chaos control in Random boolean networks," *Journal of Physics A: Mathematical and General*, vol. 31, pp. 1533–1537, 1998.

[11] H. Flyvbjerg and N. J. Kjaer, "Exact solution of Kauffman model with connectivity one," *J. Phys. A: Math. Gen.*, vol. 21, pp. 1695, 1988.

[12] U. Bastola and G. Parisi, "The critical line of Kauffman networks," *J. Theor. Biol.*, vol. 187, pp. 117, 1997.

[13] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed nets," *Journal of Theoretical Biology*, vol. 22, pp. 437–467, 1969.

[14] S. A. Kauffman, *The Origins of Order: Self-Organization and Selection of Evolution*, Oxford University Press, Oxford, 1993.

[15] J. E. S. Socolar and S. A. Kauffman, "Scaling in ordered and critical random Boolean networks," http://arXiv.org/abs/cond-mat/0212306.

[16] U. Bastola and G. Parisi, "The modular structure of Kauffman networks," *Phys. D*, vol. 115, pp. 219, 1998.

[17] A. Wuensche, "The DDlab manual," 2000, http://www.cogs.susx.ac.uk/ users/andywu/man_ contents.html.

[18] Sven Bilke and Fredrik Sjunnesson, "Stability of the Kauffman model," *Physical Review E*, vol. 65, pp. 016129, 2001.

[19] E. Dubrova, M. Teslenko, and A. Martinelli, "Kauffman networks: Analysis and applications," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, November 2005.

[20] R. Bryant, "Graph-based algorithms for Boolean function manipulation," *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, pp. 677–691, August 1986.

[21] Henrik Flyvbjerg, "An order parameter for networks of automata," *J. Phys. A: Math. Gen.*, vol. 21, pp. L955, 1988.

[22] U. Bastola and G. Parisi, "Relevant elements, magnetization and dynamic properties in Kauffman networks: a numerical study," *Physica D*, vol. 115, pp. 203, 1998.

[23] E. Dubrova and M. Teslenko, "Compositional properties of Random Boolean Networks," *Physical Review E*, vol. 71, pp. 056116, May 2005.

[24] J. A. De Sales, M. L. Martins, and D. A. Stariolo, "Cellular automata model for gene networks," *Physical Review E*, vol. 55, pp. 3262–3270, 1997.

[25] S. H. Strogatz, "Exploring complex networks," *Nature*, vol. 410, pp. 268–276, 2001.