

Threshold behavior in a Boolean Network model for SAT

Alejandro Bugacov, Aram Galstyan and Kristina Lerman
Information Sciences Institute
Univ. of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292
{bugacov,galstyan,lerman}@isi.edu

February 21, 2003

Abstract

Boolean satisfiability (SAT) is the canonical NP-complete problem that plays an important role in AI and has many practical applications in Computer Science in general. Boolean networks (BN) are dynamical systems that have recently been proposed as an algorithm for solving SAT problems [7]. We have carried out a detailed investigation of the dynamical properties of BN corresponding to random SAT problems of different size. We varied the problem size by changing the number of variables and the number of clauses in the Boolean formula. We show that dynamics of BN corresponding to 3-SAT problems display a threshold-like behavior, although this transition occurs far below the well known phase transition in the computational complexity of random 3-SAT. This threshold behavior does not appear to be connected to the transition between frozen and chaotic dynamics regimes of random BN.

1 Introduction

Boolean satisfiability (SAT) plays a fundamental role in modern problem-solving techniques. Most computational problems arising from real-world applications such as planning, scheduling, resource allocation, protein folding, computer chip verification and routing can be encoded as SAT problems. Some of the most advanced solvers for these problems are based in SAT encodings that when used in combination with state-of-the-art SAT engines [8] can handle large-scale instances very efficiently. Since all NP-complete problems can be encoded into SAT, the study of SAT problems have traditionally occupied a central role in theoretical computer science. However, due to the strong similarities with disordered systems, in recent years SAT has captured the attention of many scientist in statistical physics and mathematics and they have produced very efficient new algorithms and helped expand the understanding of the complexity of SAT [6].

SAT is a logical problem formed by Boolean variables (a, b, c, \dots) and logical constraints (AND, OR, NOT) on the variables. The SAT problem consists of finding the right assignments (True or False) for each variable such that all constraints are satisfied at the same time. SAT problems are usually expressed as a Boolean formula in conjunctive normal form (CNF). The CNF formula is formed by a conjunction of clauses. A clause is a disjunction of literals. (A literal is a variable or its negated value.) If each clause is formed by k literals then we refer to this problem as k -SAT. A 2 clauses and 5 variables 3-SAT example would be,

$$f = (a \vee b \vee \bar{c}) \wedge (b \vee \bar{d} \vee e) \quad (1)$$

It is a well known phenomenon that random k -SAT undergoes a phase transition in computational complexity for $k > 2$. This transition occurs for specific values of $\alpha = M/N$, the ratio of number of clauses (M) to number of variables (N). For low values of α the problem is easy to solve and there is a large number of possible valuations of the variables that solve the problem, in this regime we say that the problem is under-constrained. For large values of α , the problem is over-constrained and it is easy to discover early on the search process that no solution to the problem exists. However, around the critical value α_c , the problem

becomes computationally hard [8, 3] and sometimes there is only a single valuation, out of the 2^N possible combinations, that solves the formula.

In the case of *random* 3-SAT, it is well known that $\alpha_c \approx 4.25$. However, the fact that an ensemble of random formulas with a given value of α will show a phase transition in computational complexity is insufficient to predict the behavior of a single instance of the problem. The problems derived from real world-applications are usually not extracted from random distributions and the effect of the structure of the problem on the predicted value of transition behavior is still an open research issue.

1.1 Boolean Networks

A Boolean Network (BN) is a directed graph of nodes, where each node is characterized by a boolean value x_i and a boolean function F_i , that defines the dynamics of the network. Each node takes as input the values of its neighbors at time t and maps it to a new value at $t + 1$ according to F_i . In Random Boolean Networks (RBN) both the neighbor nodes and the boolean functions are chosen randomly. RBN-s were proposed by Kauffman [4] as a model of genetic regulatory networks, and have since been used to model complex adaptive systems, including multi-agent systems [2].

We borrow concepts from dynamical systems theory to describe the state space of random BN. Each configuration or state of the network of N nodes may be considered as a point in a highly dimensional state space, where each dimension corresponds to one of nodes. Evolution of the network in time from some initial state traces a trajectory through state space. If, after some period of time, the network comes back to one of the states it already visited L steps in the past, it will forever retrace this sequence of L states. We say that the system has fallen into a *period L attractor*. The states in the trajectory leading up to the attractor (the transient) are said to lie in the *basin* of attraction. Attractors of length $L = 1$ are called *fixed points* of the network.

The dynamics of a RBN is fully characterized by the average network connectivity K and the so called homogeneity parameter $0.5 \leq p \leq 1$ which determines the majority fraction of 0's and 1's in the output of the boolean functions. Let us denote $\gamma = 2Kp(1 - p)$. Then for $\gamma > 1$ the dynamics of the system is chaotic with an exponentially increasing length of attractors as the system size grows, while for $\gamma < 1$ the network reaches a frozen configuration. The case $\gamma = 1$ corresponds to a phase transition in the dynamical properties of the network [1, 5] and is often referred as the edge of the chaos [1, 9].

Because they describe “networks of influence” (value a node can take depends on values of its neighbors), BN can serve as models of SAT. Indeed, in a recent paper Milano and Roli described a BN based algorithm for SAT problems [7]. They found, however, that such a deterministic algorithm performs very poorly compared to other alternatives, such as *GSAT*. They also considered asynchronous BN (nodes update their states asynchronously), and stochastic BN (a node does not update its state with a certain probability), and showed that these modifications can significantly improve the performance of algorithm.

In this paper we report on our investigation of the dynamical properties of deterministic BN algorithm proposed in [7]. In particular, we studied how the performance of the algorithm depends on the clause to variable ratio α . We varied the problem size by changing both the number of variables and the number of clauses in the Boolean formulas, as well as the number of variables in each clause ($k = 2, 3$). We discovered that dynamics of BN corresponding to random 3-SAT problems display a threshold behavior for $\alpha \approx 2.0$, which is far below the well known phase transition in computational complexity of 3-SAT. Remarkably, this threshold behavior becomes more dramatic as one increases the problem size, a feature which is characteristic for a phase transition.

2 BN as an Algorithm for Solving SAT Problems

Milano and Roli [7] have proposed a mapping that transforms a SAT problem into a BN where every node of the network corresponds to a boolean variable in the SAT formula. They proved this mapping is *complete* and *sound*, meaning that there is a one-to-one correspondence between solutions of SAT instances and fixed points of the network. The mapping is defined as follows: Given a boolean formula with N variables in CNF: $\Phi = c_1 \wedge c_2 \wedge \dots \wedge c_m$

For each variable $x_i, i = 1, \dots, N$

Let $O_i = c_j, j = 1, \dots, m | x_i \in c_j$, and $A_i = c_j, j = 1, \dots, m | \bar{x}_i \in c_j$

Boolean function $F_i \equiv ((x_i \wedge \text{And}(A_i)) \vee \overline{\text{And}(O_i)})$

where $\text{And}(X)$ is the logical operator \wedge applied to the elements of X , and overline denotes negation.

The algorithm for finding a solution of a SAT instance is as follows: starting at a random initial state, the network evolves until either an attractor of length L is reached or maximum number of allowed steps T_{max} is exceeded. If the network relaxes to a fixed point, it is returned as a solution: otherwise the algorithm restarts from another random state. The maximum number of restarts is limited. Milano and Roli noted that the deterministic BN significantly underperforms local search algorithms like GSAT in most parameter ranges. In fact, BN are only somewhat better than the “generate and test” approach to solving SAT. The improvement comes from the fact that unlike in “generate and test”, the initial state does not have to be a solution — as long as it is in the basin of a fixed point, BN will eventually evolve to this fixed point. Despite the disappointing results, we were interested in further studying the properties of BN and investigating a possible connection between phase transition in the dynamics of random BN and phase transition in the computational complexity of SAT.

2.1 Experimental Results

We carried out a detailed analysis of the solution space of 3-SAT problems of varying sizes. We generated random instances of 3-SAT problems with N variables and M clauses. For small problem sizes, up to $N = 20$, we were able to exhaustively scan the state space of the BN, but for larger problems, we randomly sampled the state space by evolving the network from 2^{20} random initial states.

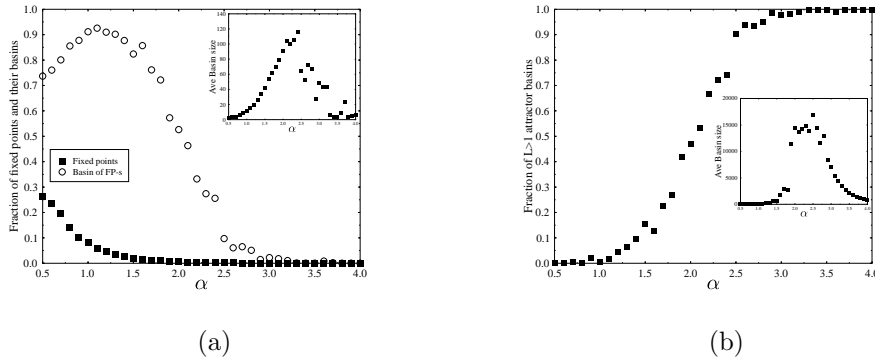


Figure 1: (a) Fraction of fixed points and the weight of their basins of attraction and (b) the weight of the basins for $L > 1$ attractors vs α for 20 variable 3-SAT problems. The insets show the corresponding average basin sizes.

2.1.1 State Space

Figure 1(a) shows the fraction of fixed points and the weight of their basins (i.e., the fraction of points in the basins) vs $\alpha = M/N$ for $N = 20$ 3-SAT problem set. Each point is an average of 100 independent trials. The fraction of fixed points (solutions) sharply drops to zero with α ; however, the total size of the basins of attraction of fixed points remains significant as α increases. Therefore, the probability of finding a solution, which is the probability a randomly chosen state will be either a fixed point or lie in the basin of attraction of one (for sufficiently large T_{max}) remains high up to $\alpha \approx 2.0$.

The reason that the probability of finding a solution does not drop at the same rate as the number of fixed points is due to the fact that the weight of the basins of the fixed points grows with α initially, as shown in the inset of Fig. 1(a) for random 3-SAT formulas with $N = 20$. The basin size attains maximum value around $\alpha \approx 2$, and sharply drops after that. For $\alpha > 3.5$ the basins shrink until the state space consists

of a small number of fixed points without a basin of attraction. In the overconstrained regime where SAT formulas generally do not have any solutions, the fixed points disappear entirely. Instead, in this regime, the dynamics of BN is characterized by attractors, mostly of period $L = 2, 4$, but also a small fraction higher period attractors. More importantly, we found that this $L > 1$ attractors are characterized by huge basins, which for $\alpha > 3$ constitute to more than 95% of the whole state space as shown in Fig. 1(b). This results in very long transients in the network dynamics, and hence, in a very poor performance of the algorithm.

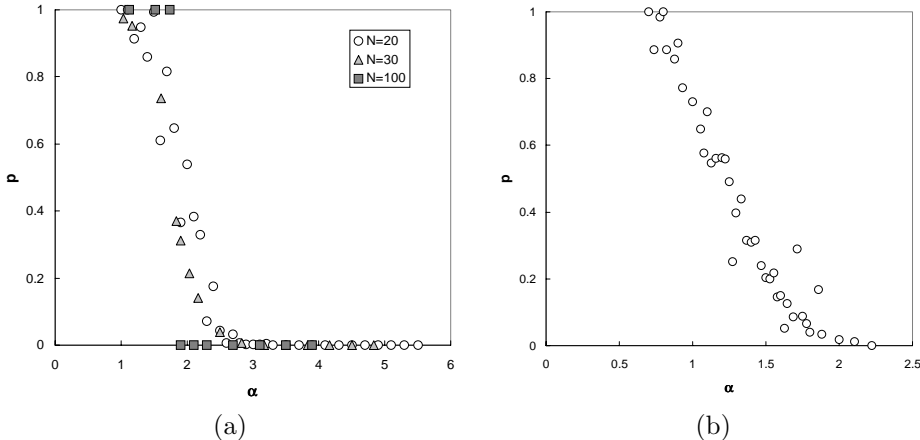


Figure 2: (a) Probability of finding a solution vs α for three different 3-SAT problem sizes. (b) Probability of finding a solution vs α for 2SAT problems with $N = 20$.

Figure 2(a) shows the probability of finding a solution vs α for problem sizes $N = 20, 30, 100$. For $N = 20$, this probability was computed as the fraction of states that were found to belong either to fixed points or their basins. For larger problem sizes, solution probability was defined as a ratio of the number of states that belong to the fixed points or their basins and the total number of states visited by the algorithm from the 2^{20} initial states. Of course, for $N = 100$ we were able to sample only a small fraction of the total state space.

Note that solution probability displays threshold behavior with respect to α reminiscent of the phase transition in computational complexity that has been observed for 3-SAT. Moreover, the threshold becomes sharper as the network size increases. Indeed, such sharpening of threshold is the hallmark of phase transitions. We note that this threshold is observed for $\alpha \approx 2$, far below the phase transition in computational complexity, estimated for 3-SAT to occur at $\alpha_c \approx 4.3$.

As we already mentioned, the threshold behavior is caused by diverging transients around $\alpha \approx 2.0$. This is already visible for small problem sizes (Fig. 1 for $N = 20$) and becomes much more pronounced for larger problems. As basins grow as $\alpha \rightarrow 2$, so does the time to find the solution (fixed point). Note also that the average basin size can be used to set an upper bound on the number of iterations used to find a solution.

2.1.2 Comparison with WalkSAT

In addition to studying solution space of BN, we have also undertaken a qualitative comparison with WalkSAT, a popular algorithm for solving SAT problem instances. Unlike BN, WalkSAT is a non-deterministic algorithm. In our trials, we had both WalkSAT and BN try to find a solution starting from the same initial state. For $\alpha < \alpha_c$, WalkSAT generally found a solution after a dozen or so steps (flips). BN, on the other hand, took many more iterations to converge to a fixed point, if at all. Table 1 presents results for WalkSAT performance on a set of 20 randomly generated 3-SAT problems with $\alpha = 1.50$ and $\alpha = 4.55$. The solutions found by both algorithms were different, indicating that WalkSAT trajectory switches between basins of BN. It is remarkable that WalkSAT manages to find a solution for $\alpha = 4.55$ within a few hundred flips, when the probability that a solution exists is a small fraction of a percent.

α	WalkSAT flips	p
1.50	7.12 ± 3.31	0.88
4.55	309.74 ± 105.43	2.56×10^{-5}

Table 1: Comparison of WalkSAT and BN performance on a set of $N = 20$ problems for two different values of α

2.1.3 2-SAT

We have also studied BN corresponding to 2-SAT problems. 2SAT formula have exactly two boolean variables in each clause. These problems are known to have a satisfiability/unsatisfiability transition at $\alpha = 1.0$. Fig. 2(b) shows the probability of finding a solution (fraction of total states that are either fixed points or their basins). Each point is an average of 100 independent trials. Although the threshold is quite broad, as would be expected for such relatively small problem sizes, it is consistent with the $\alpha = 1.0$ transition.

2.2 Dynamical Properties of k-SAT Networks

As we mentioned above, the BN algorithm for the k-SAT demonstrates a threshold behavior: for small α the algorithm finds the solution (fixed point) after a relatively small number of steps, and for large enough α the network settles into a usually short-period attractor. For α close to 2 (which corresponds to connectivity $K \approx 3\alpha \sim 6$), however, the network dynamics has long transients, whose length, we believe, increase exponentially with the size of the system. Moreover, our simulations for large system sizes suggest that long transients at threshold belong to relatively short period attractors. Note that this behavior is very different from the frozen/chaotic phase transition in RBN, where the chaotic phase is characterized by attractors with exponentially large (in N) periods. Although a full consideration of this difference goes beyond the scope of this paper, below we provide some insights on its origin.

Note that the boolean function for variable x_i in the k-SAT network strongly depends on the value of x_i itself. It is useful to stress this dependence explicitly. One can represent the dynamics of node x_i by: $x_i(t+1) = x_i(t)f_i(A_i) + (1 - x_i(t))g_i(O_i)$, where A_i (O_i) is the set of clauses in the boolean formula that contain \bar{x}_i (x_i) as defined in Section 2, and $f_i(A_i) = \text{And}(A_i)$, $g_i(O_i) = \overline{\text{And}(O_i)}$. For large system sizes the average number of clauses in A_i and O_i is $n \sim \frac{3}{2}\alpha$. If we neglect the probability of having a ‘‘multiple connection’’ to the same node, which is justified for large system sizes, then it is easy to see that the number of 1-s in the output of the function $f_i(A_i)$ decays exponentially as $(3/4)^n$. Similarly, the fraction of inputs for which $g_i(O_i)$ returns 0 decays as $(3/4)^n$. Hence, if one assumes that the inputs are assigned randomly at each time step (annealed approximation), then the probability for a node to change its state exponentially approaches to 1 with α . Consequently, for sufficiently large α the dynamics of the network will consist of period two attractors, where each node constantly changes its state. Indeed, our simulation confirm this observation. In contrast, for a RBN with large α (large connectivity K) the dynamics should be strongly chaotic (note that the homogeneity coefficient p is close to 1/2 because of the symmetry in updating rule). This difference is due to the fact that the boolean functions in the k-SAT network are not random, but rather have a structure, which keeps the system off the chaotic regime for large K .

3 Conclusion

We have studied Boolean Networks as an algorithm for finding solutions to random k-SAT problems. Our intent was to (i) make a connection between the well-understood dynamic properties of random BN and the computational complexity of 3-SAT problems, and (ii) use additional topological features (such as network connectivity, clustering coefficient) to predict the computational complexity of individual problem instances. We have carried out an intensive investigation of the dynamical properties of BN corresponding to k-SAT formulas of different sizes. Although we were not successful in our original goals, we discovered that dynamics of BN corresponding to 3-SAT problems display a threshold behavior. Moreover, the threshold grew sharper

as the problem size was increased. We observed this threshold at $\alpha \approx 2.0$, far below the well known phase transition in computational complexity of 3-SAT, that occurs around $\alpha = 4.3$. There also does not appear to be any connection between either the observed threshold at $\alpha = 2.0$ or the phase transition in computational complexity of 3-SAT and the phase transition in the dynamics of random BN. We believe that even though the 3-SAT problem instances are random, the resulting BN have structure different from random BN.

4 Acknowledgments

Authors would like to thank Tad Hogg and Bart Selman for their encouragement and useful comments. The research reported here was supported in part by the Defense Advanced Research Projects Agency (DARPA) under contracts number F30602-00-2-0573, in part by the National Science Foundation under Grant No. 0074790.

References

- [1] B. Derrida and Y. Pomeau. Random networks of automata: A simple annealed approximation. *Europhys. Lett.*, 1:45, 1986.
- [2] Aram Galstyan and Kristina Lerman. Adaptive boolean networks and minority games with time-dependent cutoffs. *Physical Review E*, 2002.
- [3] Tad Hogg, Bernardo A. Huberman, and Colin Williams, editors. *Frontiers in Problem Solving: Phase Transitions and Complexity*, volume 81 of *Special issue of Artificial Intelligence*. march 1996.
- [4] Stuart A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22:437–467, 1969. Boolean Networks.
- [5] Stuart A. Kauffman. *The Origins of Order*. Oxford University Press, 1993.
- [6] Marc Mezard and Riccardo Zecchina. The random k-satisfiability problem: from an analytic solution to an efficient algorithm. *Phys. Rev.*, E 66:056126, 2002.
- [7] M.Milano and A.Roli. Solving the satisfiability problem through boolean networks. In *Lecture notes in Artificial Intelligence, Vol. 1792*, New York, 2000. Springer.
- [8] Bart Selman, Henry Kautz, and B. Cohen. Local search strategies for satisfiability testing. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 26:521–532, 1996.
- [9] R. V. Sole, B. Luque, and S. Kauffman. Phase transitions in random networks with multiple states. working paper 00-02-11, Sante Fe Institute, 2000.