Parallel complexity of random Boolean circuits

J Machta[1,2], S DeDeo[1], S Mertens[1,3] and C Moore[1,4,5]

[1] Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501, USA
[2] Physics Department, University of Massachusetts, Amherst, MA 010003, USA
[3] Institut für Theoretische Physik, Otto-von-Guericke Universität,
D-39106 Magdeburg, Germany
[4] Computer Science Department, University of New Mexico, Albuquerque,
NM 87131, USA
[5] Department of Physics and Astronomy, University of New Mexico,
Albuquerque, NM 87131, USA
E-mail: machta@physics.umass.edu, simon@santafe.edu, mertens@ovgu.de and
moore@santafe.edu

Abstract. Random instances of feedforward Boolean circuits are studied both analytically and numerically. Evaluating these circuits is known to be a **P**-complete problem and thus, in the worst case, believed to be impossible to perform, even given a massively parallel computer, in a time much less than the depth of the circuit. Nonetheless, it is found that, for some ensembles of random circuits, saturation to a fixed truth value occurs rapidly so that evaluation of the circuit can be accomplished in much less parallel time than the depth of the circuit. For other ensembles saturation does not occur and circuit evaluation is apparently hard. In particular, for some random circuits composed of connectives with five or more inputs, the number of true outputs at each level is a chaotic sequence. Finally, while the average case complexity depends on the choice of ensemble, it is shown that for all ensembles it is possible to simultaneously construct a typical circuit together with its solution in polylogarithmic parallel time.

Contents

1. Introduction

One of the most fruitful areas of interaction between statistical physics and computer science is the study of phase transitions in random instances of computational problems. Using ideas from statistical physics, a number of studies have demonstrated that, for many **NP**-hard optimization problems, the running time of algorithms for solving these problems is largest at a phase transition separating a regime where a solution exists from a regime where no solution exists [1]–[3].

It has also been found that, for some random ensembles of hard problems, if the problem is sufficiently unconstrained, then we can sample from the distribution of (instance, solution) pairs, at least approximately, using the so-called planted ensemble [4, 5]. That is, rather than choosing a random problem and then solving it, we first choose a random solution and then choose randomly from among the instances consistent with it. This approach is especially relevant to statistical physics, since we are more interested in averaging over ensembles of disordered systems than in solving individual hard problems.

While considerable effort has been expended on understanding random ensembles of **NP**-hard problems, we are unaware of comparable investigations lower in the complexity hierarchy. In this paper we investigate both phase transitions in complexity and sampling (instance, solution) pairs for random ensembles of the circuit value problem, a problem in the class **P**.

Contained within the class **P** of problems that are solvable in polynomial time there are several nested hierarchies of complexity classes that are best understood in terms of

parallel computation [6]. For our purposes, the most important of these classes is **NC**, the class of problems that can be solved in parallel in polylogarithmic time, i.e. $\mathscr{O}(\log^k N)$ time for some constant k, where N denotes the size of the problem. In the definition of **NC** the model of computation that is considered is the PRAM, an idealized parallel computer with a number of processors that is allowed to scale polynomially in the size of the problem and with a global random access memory through which any pair of processors can communicate in $\mathscr{O}(1)$ time. The PRAM runs synchronously and each processor runs the same program but has a distinct label so that it may carry out distinct computations. In each time step, each processor may read or write to a global memory cell. Conflicts that arise if two processors attempt to write to the same memory cell at the same time may be resolved in different ways, but these differences do not change the definition of the class **NC**. Although we have just defined **NC** in terms of a specific model of parallel computation, it is a quite robust class of problems and can be equivalently defined in terms of families of Boolean circuits, alternating Turing machines and even, without reference to computation at all, in terms of properties of the first-order formal logic description of the problem [7].

Problems in **NC** include adding N numbers, multiplying two $N \times N$ matrices and finding the connected components of a graph with N vertices. In the case of addition, the parallel algorithm consists simply of pairwise addition of N numbers by $N/2$ processors, followed by pairwise addition of these partial sums by $N/4$ processors, and so on until the sum is obtained after $\log_2 N$ parallel steps.

A question then arises as to whether every problem in **P** can be solved in parallel in polylogarithmic time—that is, whether $\mathbf{P} = \mathbf{NC}$. It is widely believed, but not yet proved, that this is not the case and that there are problems in **P** that are hard to solve in parallel. Just as **NP**-complete problems are the hardest problems in the class **NP**, there is a class of **P**-complete problems that are the hardest problems in **P** to solve in parallel. Assuming that $\mathbf{P} \neq \mathbf{NC}$, **P**-complete problems are inherently sequential—they cannot be solved in polylogarithmic time on a PRAM with polynomially many processors.

The canonical **P**-complete problem is the circuit value problem (CVP). An instance of CVP is specified by a feedforward Boolean circuit with given truth values for the inputs, and the problem is to find the outputs. If the circuit has N gates, then this problem is clearly in **P** since we can evaluate the output of each gate in roughly $\mathscr{O}(N)$ time. The question is to what extent we can parallelize this computation. The depth of a circuit is the longest path from an input to an output. By evaluating all the gates in a given layer simultaneously, then we can solve CVP in an amount of parallel time proportional to the depth. However, it is not clear that we can improve significantly on this, and since a circuit with N gates could have, say, $\sqrt{N}$ depth and $\sqrt{N}$ width, it seems unlikely that we can solve CVP in $\mathscr{O}(\log^k N)$ parallel time. Any such algorithm would have to 'skip over' many of the layers of a circuit, rather like predicting the future state of a system without having to simulate it step by step.

CVP plays the same central role in the theory of **P**-completeness as satisfiability plays in the theory of **NP**-completeness. Reductions from CVP or satisfiability are the standard tools for proving other problems **P**-complete or **NP**-complete, respectively. Just as a polynomial-time algorithm for satisfiability would imply that $\mathbf{P} = \mathbf{NP}$, if CVP is in **NC** then $\mathbf{P} = \mathbf{NC}$ and all problems that can be solved in polynomial time can be efficiently parallelized.

In this paper we demonstrate the existence of hard and easy phases and transitions between them in random ensembles of CVP.

We show that random monotone CVP is easy if the circuit almost surely saturates quickly to one of the two stable fixed points—either all TRUE or all FALSE. Random NOR CVP quickly saturates to a period-two oscillation between TRUE and FALSE. Our analysis uses simple, exact recursion relations for the expected number of TRUE outputs at one level as a function of the fraction at the previous level. Similar methods were employed by Valiant [8] to demonstrate the existence of monotone circuits that quickly evaluate the majority function. In addition to circuits composed of simple gates (i.e. AND, OR, NOR and NAND gates) we also consider random circuits built out of more complicated Boolean functions or connectives. Here we find examples where the fraction of TRUE outputs of each level of the circuit is a chaotic sequence; this leads to a qualitatively different way that CVP can become hard. Finally, we show that for any choice of parameters it is possible to simultaneously construct problem instances and their solutions quickly in parallel. It is thus easier to sample (instance, solution) pairs than to be given an instance and then solve it.

Random Boolean circuits have a long history. Early work by von Neumann [9] and Moore and Shannon [10] focused on circuits with unreliable gates. Recursion relation methods were developed in [10] that parallel the methods used here. Later work by Valiant [8] used random monotone circuits to compute the majority function. This work was extended to study other functions that can be computed by random circuits generated via a growth process, see, for example, [11] and references therein. In [11] the recursion relation is called a characteristic polynomial and the elementary Boolean function is referred to as the 'connective', a term we reserve here for more complicated Boolean functions composed of multiple AND, OR, NAND or NOR gates. Recent work by [12] considers the joint problem of random circuits and unreliable gates, finding associations between reliability bounds and macroscopic phase transitions. In our ensembles, each layer of the circuit is chosen simultaneously and the circuit has fixed width; for another ensemble of random circuits, where random gates are added one at a time, see [13].

This paper is organized as follows. In section 2 we introduce random ensembles of Boolean circuits. In section 3 we analyze the difficulty of solving monotone CVP as a function of the in-degree of the gates and other parameters describing the ensemble and demonstrate the existence of different phases of hardness. Section 4 considers the case of NOR CVP. In section 5 we show how chaotic behavior in circuit properties is possible for connectives of sufficiently high in-degree. Section 6 presents numerical results supporting the theoretical conclusions for monotone circuits. In section 7 we show that (instance, solution) pairs can be sampled in polylogarithmic parallel time even in the phases where solving giving random instances is hard. The paper concludes in section 8 with a summary and discussion.

2. Random circuit value problems

We define random Boolean circuits as follows. We think of them as lying on a rectangular grid of width L, although the circuit's topology is less restricted than the grid would suggest. On the top row, the L inputs of the circuit are independently chosen and take the value TRUE with probability τ_0 and FALSE with probability $1 - \tau_0$. There are L
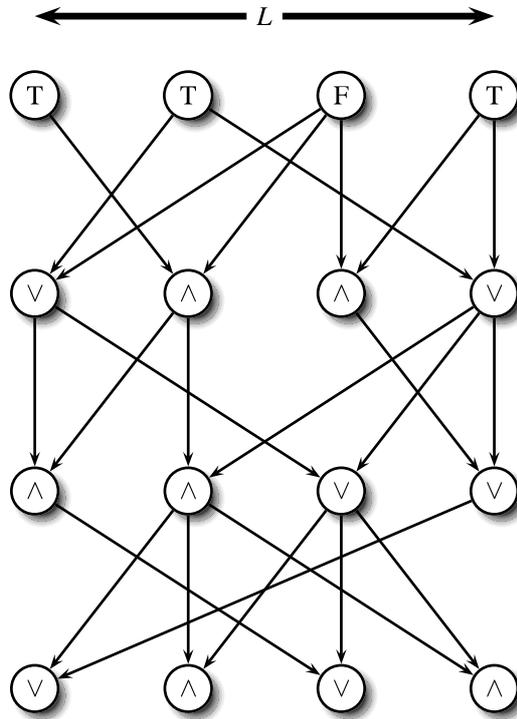
Figure 1. An example of a random monotone Boolean circuit arranged in levels on a rectangular grid of width L with each gate having $k = 2$ inputs. Each of the k inputs of a gate is attached to a randomly chosen gate at the next higher level of the circuit.

gates (or connectives) on each level, each of which has some in-degree k. Each gate on level n takes inputs from k randomly chosen gates on level $n-1$, and a gate on the first level takes inputs from k randomly chosen input values. Note that the inputs to a gate are chosen randomly with replacement so that inputs may be repeated.

We consider monotone circuits, circuits consisting entirely of NOR gates and circuits built from more complicated connectives. We start with monotone circuits, which contain only AND ($\wedge$) and OR ($\vee$) gates. Each gate is OR with probability p, or AND with probability $(1 - p)$. These gates can have any in-degree k; OR gates return TRUE if at least one of their inputs are TRUE, and AND gates return TRUE if all of their inputs are TRUE. We will explicitly study two-input and three-input gates, i.e. $k = 2$ and 3; the case $k > 3$ is very similar to $k = 3$. A typical monotone circuit with two-input gates is shown in figure 1.

Our goal is to understand the parallel complexity of the CVP problem in this ensemble, i.e. the complexity of computing the truth values on the final layer as a function of the inputs. This problem is **P**-complete whenever L is allowed to grow as some power of the total number of gates. Even monotone CVP is **P**-complete [6], since we can use De Morgan's law to push negations up through the circuit, making the gates monotone and flipping some of the gates. Since $\text{NOR}(x, y) = \neg(x \vee y)$ is a complete basis for Boolean logic, i.e. any Boolean function can be implemented using only NOR gates, NOR CVP is also **P**-complete.

3. Analysis of random monotone CVP

In monotone circuits, the homogeneous states (all TRUE or all FALSE) are absorbing states in the sense that, if they appear in one layer of the circuit, they will persist all the way to the final layer, independently of the wiring and the gates. Therefore the parallel computational complexity is determined by the time to reach one of these absorbing states.

Let τ_n denote the fraction of the gates at level n whose output is TRUE. Because of the random connectivity of the circuit, it is possible to write the expectation of τ_{n+1} exactly as a function of τ_n. For the case of monotone circuits with two inputs, $k = 2$, this is

$$\begin{aligned}
\langle \tau_{n+1} \rangle &= p(2\tau_n - \tau_n^2) + (1-p)\tau_n^2 \\
&= \tau_n + (2p-1)(\tau_n - \tau_n^2).
\end{aligned} \tag{1}$$

The first term in the first expression is the probability that an OR gate will evaluate to TRUE times the expected fraction p of OR gates, and the second term is the analogous quantity for AND gates. More generally, for k-input AND and OR gates we have

$$\langle \tau_{n+1} \rangle = R_k(\tau_n, p) \equiv p(1 - (1-\tau_n)^k) + (1-p)\tau_n^k. \tag{2}$$

In the limit of large L, τ_{n+1} is tightly concentrated, allowing us to drop the distinction between the random variable τ_{n+1} and its expectation $\langle \tau_{n+1} \rangle$. This gives a recursion relation

$$\tau_{n+1} = R_k(\tau_n, p). \tag{3}$$

The initial condition for this recursion relation is the fraction of TRUE inputs, τ_0. The endpoints $\tau = 0$ and 1 are fixed points of the recursion relations. Once the circuit saturates at $\tau = 0$ or 1, any further levels simply reproduce these truth values. Thus the depth to reach saturation is an upper bound on the parallel time required to evaluate the circuit. As we see below, this saturation depth depends on the fluctuations around $\langle \tau_{n+1} \rangle$ for finite L.

3.1. Two-input gates

Consider the case $k = 2$. The only fixed points are $\tau = 0$ and 1. From the second equality in equation (1), it is clear that the behavior of the recursion relations changes as a function of p at $p = 1/2$. For $p < 1/2$, $\tau = 0$ is stable and $\tau = 1$ is unstable, and the flow is from $\tau = 1$ to 0. The opposite holds for $p > 1/2$.

We can draw the following qualitative conclusions from these flows. A sufficiently deep random circuit with a preponderance of AND gates will almost always evaluate to FALSE, while one with a preponderance of OR gates will almost always evaluate to TRUE. Let us define the saturation depth D as the mean level n at which the circuit first saturates (almost always TRUE for $p > 1/2$ and FALSE for $p < 1/2$). We can estimate D as the least n for which $\tau_n \approx 1/L$.

First, consider the case $p < 1/2$. Linearizing around the relevant fixed point at $\tau = 0$, we obtain $\tau_{n+1} = 2p\tau_n$ and so $\tau_n = (2p)^n \tau_0$. This yields the estimate

$$D \sim \frac{\ln L}{-\ln(2p)}. \tag{4}$$

A similar calculation for $p > 1/2$, obtained by linearizing around the fixed point at $\tau = 1$, yields

$$D \sim \frac{\ln L}{-\ln(2(1-p))}.\tag{5}$$

In either case, D diverges like $1/|p - 1/2|$ as $p \to 1/2$.

The case $p = 1/2$ cannot be understood in terms of the recursion relation (1) for the expected fraction of TRUE gates. Instead of recursion relations for the expectation τ_n, we need to follow the stochastic behavior of T_n, the actual number of TRUE gates at level n. Including fluctuations in the fraction of gates of each type on a given level, the distribution of T_{n+1} is a single binomial distribution whose mean is obtained from T_n using the recursion relation, equation (1), with T_n/L replacing τ_n. In the special case that $p = 1/2$, the recursion relation is the identity and

$$T_{n+1} = \mathscr{B}(L, T_n/L).\tag{6}$$

Here $\mathscr{B}(N, p)$ is a binomial random variable whose value is the number of successes in N trials with probability of success p. The initial condition for this recursion relation is $T_0 = \mathscr{B}(L, \tau_0)$.

Equation (6) describes a random walk with variable step length on a line with absorbing states at 0 and L. In order to analyze this walk we take the large L limit and replace the binomial by a normal random variable:

$$T_{n+1} = \mathscr{N}(T_n, T_n(1 - T_n/L))\tag{7}$$

where $\mathscr{N}(\mu, \sigma^2)$ is a normal random variable with mean μ and variance σ^2. Let $P(x, n)$ be the probability density for the number of TRUE gates at level n, $P(x, n)\,dx = \mathrm{Prob}[T_n \in (x, x + dx)]$. The recursion relation for P follows from equation (7) and takes the form of an integral equation with a diffusion Green's function with a spatially varying diffusion coefficient:

$$P(x, n + 1) = \int_0^L dx'\, G(x, x')P(x', n),$$

where G is the one-dimensional diffusion kernel:

$$G(x, x') = \frac{1}{\sqrt{4\pi K(x')}}\, e^{-(x-x')^2/4K(x')}$$

and $K(x)$ is the diffusion coefficient:

$$K(x) = \frac{x}{2}\left(1 - \frac{x}{L}\right).\tag{8}$$

The saturation depth D at which the gates are all TRUE or all FALSE is the mean first-passage time to the absorbing states at $x = 0$ or L. The first-passage time for a diffusion process with a diffusion coefficient that varies as in equation (8) is analyzed in [14]. The result depends on the initial condition. In our case, $P(x, 0) = \delta(x - \tau_0 L)$, where τ_0 is the expected fraction of TRUE inputs. After appropriate changes of variable to put our expression in the form given in section 4.6.2 of [14], we obtain[6]

$$D = -2L[\tau_0 \ln \tau_0 + (1 - \tau_0)\ln(1 - \tau_0)] = 2Lh(\tau_0),\tag{9}$$

---

[6] Our results follow from equation (4.6.6) of [14] as corrected in the online errata.

where $h(\tau) = -\tau \ln \tau - (1 - \tau) \ln(1 - \tau)$ is the Gibbs–Shannon entropy of the initial distribution of inputs. Note that D is now linear in the width of the circuit instead of logarithmic, as was the case for $p \neq 1/2$, suggesting that the parallel complexity is $\mathcal{O}(L)$ rather than $\mathcal{O}(\log L)$.

3.2. Three-input gates

Next, consider the case where each gate has three inputs. The fixed point structure of the recursion relations for $k = 3$ is more complicated than for $k = 2$. In addition to the fixed points at $\tau = 0$ and 1, there is a third fixed point at $\tau^* = 3p - 1$. This fixed point is meaningful only for $1/3 < p < 2/3$, since otherwise it is outside the unit interval.

For $p < 1/3$, it is straightforward to verify that $\tau_{n+1} < \tau_n$ so that $\tau = 0$ is the stable fixed point and $\tau = 1$ the unstable fixed point. Thus the regime $p < 1/3$ for three-input gates is similar to the regime $p < 1/2$ for two-input gates: in both cases the circuit almost always saturates to FALSE at depth $D = \mathcal{O}(\log L)$. For $p > 2/3$, $\tau = 1$ is the stable fixed point and the circuit saturates to TRUE, again at logarithmic depth.

The regime $1/3 < p < 2/3$ has no analogy for two-input gates. Linearizing the recursion relation around the fixed point by setting $\tau_n = \tau^* + \delta\tau_n$ in equation (3), we obtain

$$\delta\tau_{n+1} = (9p^2 - 9p + 3)\delta\tau_n. \tag{10}$$

The coefficient of $\delta\tau_n$ in this equation is between 0 and 1, implying that the fixed point $\tau^* = 3p - 1$ is stable while the fixed points at $\tau = 0, 1$ are unstable. The conclusion is that for any $1/3 < p < 2/3$ and $0 < \tau_0 < 1$ the circuit fails to saturate, but instead behaves stochastically for a very long time, with τ making $\mathcal{O}(1/\sqrt{L})$ fluctuations around τ^*. For any finite L it is possible for the system to fall into one of the absorbing states, either all TRUE or all FALSE, but at each step this is exponentially unlikely. Thus for exponentially long time scales, saturation will almost surely occur, but for depths and widths that are related polynomially the circuit will simply have to be evaluated one level at a time.

3.3. Gates with more than three inputs

The case $k > 3$ is qualitatively similar to the case $k = 3$. For $p < 1/k$, the only stable fixed point is $\tau = 0$, and for $p > 1 - 1/k$ the only stable fixed point is $\tau = 1$. In both cases, the circuit saturates at logarithmic depth.

For $1/k < p < 1 - 1/k$, on the other hand, both these fixed points become unstable, and there is a single attracting fixed point $0 < \tau^* < 1$. The recursion $R_k(\tau, p)$ rises sharply from $\tau = 0$, is relatively flat near $R_k(\tau, p) \approx p$ and then rises sharply again to one near $\tau = 1$. Thus there is one stable fixed point at $\tau^* \approx p$, and it becomes increasingly stable as k increases. The saturation depth is again exponential, and these cases of CVP are hard.

4. Analysis of random NOR CVP

The defining feature of the monotone ensembles considered above is the absence of negation: increasing the fraction of TRUE inputs increases the likelihood of TRUE

outputs. In this section we consider random ensembles with non-monotone gates, namely NOR CVP. As mentioned above, NOR CVP is **P**-complete, since NOR is a complete basis for Boolean logic.

We define a random ensemble of NOR gates on a grid of width L. Each gate takes k inputs, chosen randomly with replacement, from the level above it. Initially, there is a fraction τ_0 of TRUE inputs. Since a NOR gate returns TRUE if and only if all its inputs are false, the recursion relation is

$$\tau_{n+1} = S_k(\tau_n) \equiv (1 - \tau_n)^k. \tag{11}$$

First, consider the case $k = 2$. The recursion relation has a single fixed point in the unit interval at $\tau^* = (3 - \sqrt{5})/2 \approx 0.382$. Since $|\mathrm{d}S_2(\tau^*)/\mathrm{d}\tau| = \sqrt{5} - 1 > 1$, this fixed point is unstable and flows to a stable period-two orbit, oscillating between $\tau = 0$ and 1. This orbit is stable, since $S(S(\tau)) = 4\tau^2 + \mathscr{O}(\tau^3) < \tau$ for sufficiently small τ.

Once the circuit reaches this fixed point, it has saturated in a way analogous to the monotone ensemble of the previous section. The gates alternate between all TRUE and all FALSE, and no additional computation is needed to predict the behavior deeper in the circuit. Since the period-two orbit is approached exponentially, and saturation occurs when $\tau_n \approx 1/L$, the saturation depth is $D \sim \log L$ just as in the easy regime of monotone CVP. This is the case even if τ_0 starts at the unstable fixed point τ^*, since random fluctuations cause τ to vary by $\mathscr{O}(1/\sqrt{L})$ in any case. The expected distance from τ^* grows exponentially, driving τ a distance $\mathscr{O}(1)$ away from τ^* after $\mathscr{O}(\log L)$ steps. Thus the entire phase diagram for NOR CVP is easy.

Unlike monotone CVP, the case $k = 2$ is generic for NOR CVP. For any k the period-two orbit is stable since $S_k(S_k(\tau)) = (k\tau)^k + \mathscr{O}(\tau^{k+1}) < \tau$ for τ sufficiently small. Since $S_k(x)$ is a convex function that varies between one at $x = 0$ and zero at $x = 1$, there is exactly one fixed point τ^* in the unit interval; it is unstable for all k and its value tends toward zero as k increases. Apart from these quantitative differences, the behavior of k-input random NOR circuits is the same for all k. The saturation depth is logarithmic, and predicting the behavior of the circuit is easy for all k and all τ_0.

5. Chaotic connectives

While the recursion relations equations (3) and (11) corresponding to simple AND, OR and NOR gates have simple dynamics, with a handful of fixed points and stable periodic orbits, connectives that implement more complicated Boolean functions can produce a variety of complicated behaviors. In this section we show an example of such a connective whose dynamics are chaotic, requiring a step-by-step simulation even to keep track of the expected fraction of TRUE outputs.

Since the circuit is wired randomly, many connectives—those whose truth table entries can be related by permutations of input bits—produce equivalent recursion relations. For a connective of in-degree k, the full space of recursion relations is thus covered by $k + 1$ integer parameters, $0 \le \alpha_i \le \binom{k}{i}$, which define how many input bit configurations with i TRUE bits lead to a TRUE output. The recursion relation for such a connective is

$$\tau_{n+1} = \sum_{i=0}^{k} \alpha_i \tau_n^i (1 - \tau_n)^{k-i} \equiv F(\tau_n). \tag{12}$$

If the layers consist of a mixture of different connectives with the same in-degree k, as in the mix of AND and OR gates studied in section 3, the α_i become weighted averages and can take arbitrary real values between 0 and $\binom{k}{i}$.

Varying the α_i lets us construct connectives, or mixtures of connectives, such that the recursion equation (12) becomes a wide variety of functions $F(\tau)$ on the unit interval. For instance, consider the logistic map,

$$f(\tau) = r\tau(1 - \tau). \tag{13}$$

This undergoes a series of period-doubling transitions as r increases and becomes chaotic at $r \approx 3.57$ (see, e.g., [15]). Setting equation (12) equal to equation (13) and solving for α_i, we find the logistic map can be reproduced by taking

$$\alpha_i = \begin{cases} 0 & \text{if } i = 0 \quad \text{or} \quad i = k, \\ r\dbinom{k-2}{i-1} & \text{if } 0 < i < k. \end{cases} \tag{14}$$

Connectives of this form are non-monotone and XOR-like, peaking at $\tau = 1/2$. Note that there is an absorbing state, i.e. the fixed point $f(0) = 0$.

Because of the upper bound on α_i, the maximum allowable value $r_{\max}$ of the parameter r is a function of the size of the connective. For $k = 2$ we have $r_{\max} = 2$, which corresponds to the standard XOR function. More generally, we have

$$r_{\max} = \frac{2(2\lceil k/2 \rceil - 1)}{\lceil k/2 \rceil}, \tag{15}$$

where $\lceil x \rceil$ is the smallest integer greater than or equal to x. As k increases, $r_{\max}$ increases and approaches 4, giving us the full range of logistic maps. In particular, we can reach the chaotic range $r \gtrsim 3.57$ with connectives of in-degree $k \geq 9$.

However, the binomials $\binom{k-2}{i-1}$ appearing in (14) have no common factors. Therefore, for an arbitrary r—and, in particular, for r in the chaotic regime $r \gtrsim 3.5$—the α_i cannot be integers, and reproducing the logistic equation exactly requires a mixture of at least two connectives of sufficient k.

Given this restriction, one might ask whether it is possible to achieve chaotic behavior similar to that of the logistic map with a single connective. One approach is to approximate the logistic map by rounding the α_i to nearby integers. Given the universality of the period-doubling route to chaos, if the resulting approximation is good enough, the behavior will be similar. For $k = 9$ and $r = r_{\max} = 18/5$, we can take

$$\alpha_i = \begin{cases} 0 & \text{if } i = 0 \quad \text{or} \quad i = 9, \\ \left\lceil \frac{18}{5} \dbinom{7}{i-1} \right\rceil & 0 < i < 9 \end{cases} \tag{16}$$

and iterating the map for a range of different initial conditions shows chaotic behavior. We can measure the Lyapunov exponent $\lambda = \langle \log |\mathrm{d}S/\mathrm{d}\tau| \rangle$, where averages are taken over the courses of long trajectories, and find that $\lambda \approx 0.14 > 0$. For comparison, the logistic map with $r = 18/5$ has $\lambda \approx 0.18$.

Instead of approximating the logistic map, we can look directly for chaotic behavior via an exhaustive search of all possible connectives. Since connectives whose truth tables
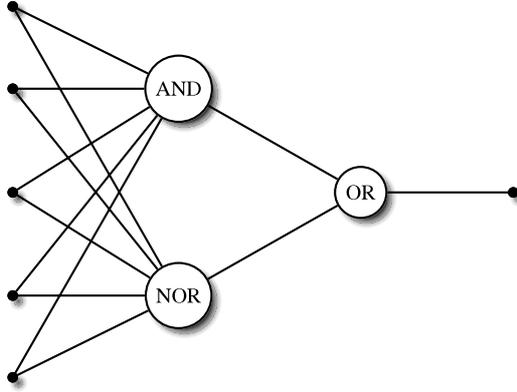
Figure 2. The logic circuit diagram corresponding to equation (18), one of the minimal arity connectives that produces chaotic behavior.

are related by a permutation of the input are equivalent, there are far fewer than 2^{2^k} connectives to check; the number of distinct connectives is

$$n(k) = \prod_{i=0}^{k} \left[\binom{k}{i} + 1 \right]. \tag{17}$$

Searching the maps associated with each connective for those with positive Lyapunov exponents, we find the first chaotic maps at $k = 5$. Of the 17 424 distinct connectives for $k = 5$, there are six chaotic ones. The simplest one (figure 2) consists of a NOR gate and an AND gate, combined with an OR gate:

$$\alpha_i = \begin{cases} 1 & \text{if } i = 0 \quad \text{or} \quad i = 5, \\ 0 & \text{if } 0 < i < 5. \end{cases}$$

This gives the recursion relation

$$F(\tau) = \tau^5 + (1 - \tau)^5, \tag{18}$$

which has a Lyapunov exponent $\lambda \approx 0.20$. It is a unimodal map in the chaotic regime, with unstable orbits of period 1, 2, 4 and so on. Its first, second and fourth iterates are shown in figure 3.

This map has an absorbing state at $F(1) = 1$, which it can fall into due to finite-size fluctuations. But, since its chaotic attractor is bounded away from 1, in the interval $[F(1/2), F(F(1/2))] = [0.0625, 0.7242]$, it will take exponential time for this to occur. Thus the saturation depth D is exponential in L and the CVP requires step-by-step simulation.

It is also possible to have combinations of connectives such that there is no absorbing state—that is, such that neither $\tau = 0$ and 1 is a fixed point, nor do they form an orbit of period two. In that case the circuit never saturates and the CVP is again hard.

6. Numerical results for two-input monotone CVP

We carried out numerical simulations of random monotone CVP to check the results obtained in section 3.1. In our simulations, L ranges from 128 to 16 384, and we take
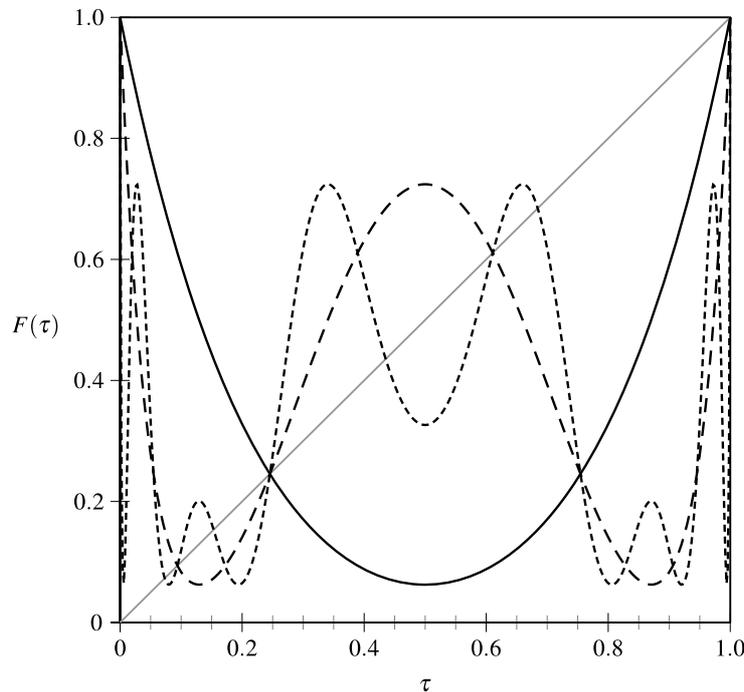
Figure 3. Iterated maps of the chaotic connective, equation (18) and figure 2, showing unstable periodic orbits of period 1 (solid line), 2 (long dashed line) and 4 (short dashed line).

1000 realizations of the circuit per data point. Recall that p is the fraction of OR gates. We first consider two-input gates with $p \neq 1/2$. Figure 4 shows the saturation depth D versus $\log L$ for several values of p. The initial fraction of TRUE inputs is $\tau_0 = 1/2$. The results demonstrate that the saturation depth increases logarithmically with the circuit width with a slope that increases as p approaches the critical value $1/2$. Figure 5 shows the slope $m = D/\log L$, revealing the divergence as $p \to 1/2$. The data fits the prediction of equation (5) very well.

Next we consider the critical case $p = 1/2$. Figure 6 shows the saturation depth D as a function of L for various initial fractions of TRUE inputs τ_0. Note that D increases linearly with L as predicted in equation (9). The slope $m = D/L$ is shown in figure 7 along with the prediction of equation (9). The data confirms our estimate based on the mean first-passage time calculation.

7. Fast sampling of evaluated random circuits

In the previous sections we showed that random instances of CVP are hard or easy to solve in parallel, depending on the types of connectives and the fraction of connectives of each type. In this section we consider the complexity of simultaneously generating a random instance of CVP, together with its solution—in other words, of sampling the distribution of (instance, solution) pairs. We show the surprising result that, for any choice of parameters and connectives, random instances of CVP and their solutions can be generated in polylogarithmic parallel time. The construction depends on generating
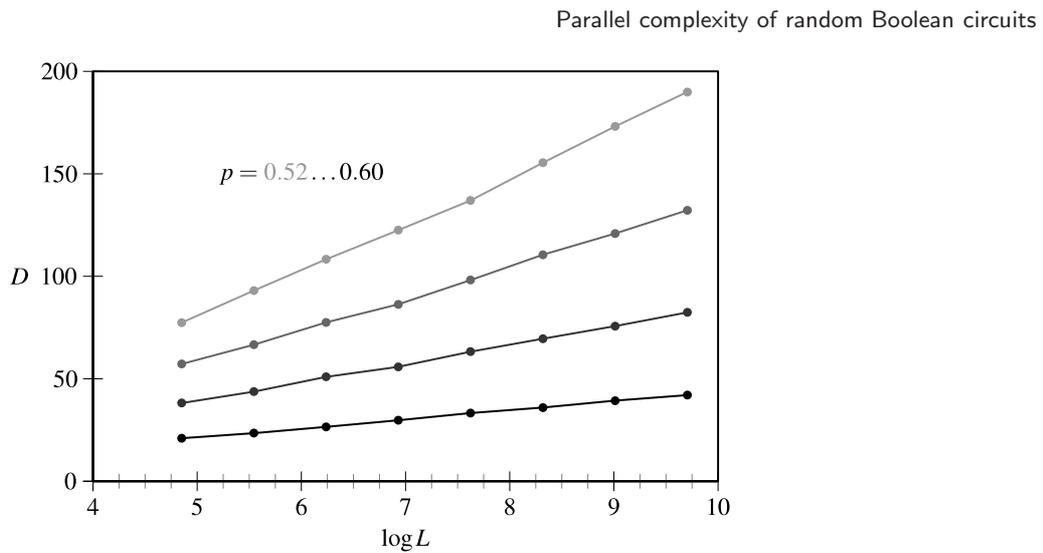
Figure 4. Saturation depth D versus the log of the circuit width L for $k = 2$ and various values of the fraction of OR gates p.
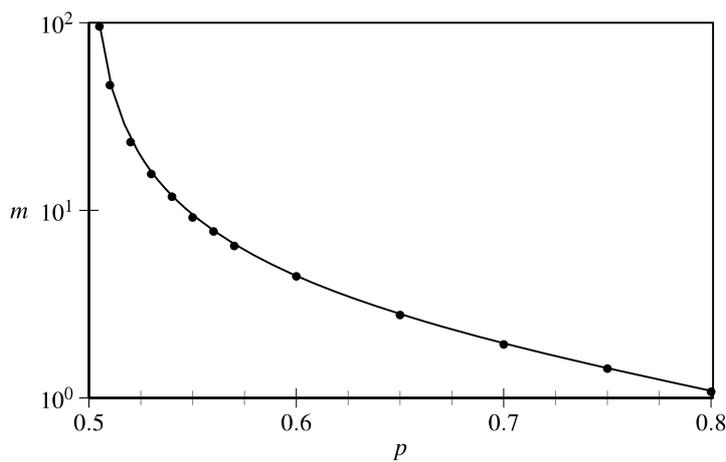

Figure 5. Slope m of the logarithmic scaling of saturation depth versus the fraction p of OR gates for $k = 2$. The solid line is the prediction of equation (5), $m = 1/\ln(2(1-p))$.

individual levels of the circuit independently in parallel and then connecting these levels together into a circuit and its solution. Each level is defined by the placement of each type of connective, the number of TRUE inputs to the level and the evaluation of each connective.

Here we sketch a polylogarithmic time PRAM program that carries out the construction of a single instance of random CVP together with its solution. The first step is to generate the inputs to the circuit, $X_0^i \in \{0, 1\}$, $i = 1, \ldots, L$ and the type of each connective on each level n, Y_n^i for $i = 1, \ldots, L$ where, for example, Y might take the value 'three-input OR' or 'five-input NAND'.

The next step in the construction is to evaluate each connective. Since we do not yet know the number of TRUE inputs T_n for level $n + 1$, we must generate the outputs of all
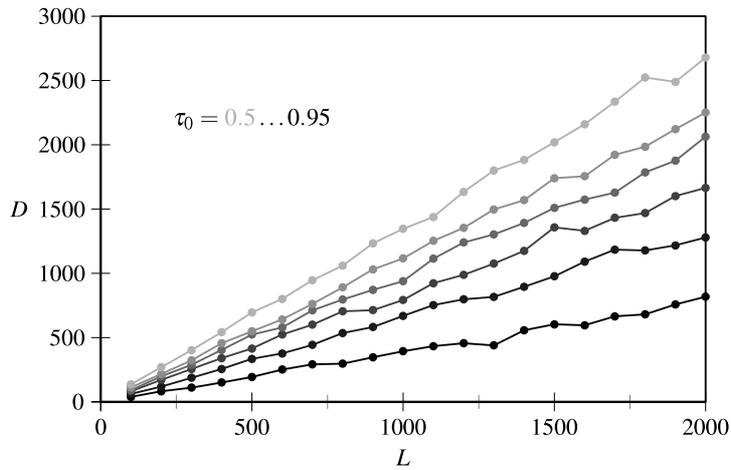
Figure 6. Saturation depth D versus circuit width L for various values of the fraction of TRUE inputs τ_0 for the critical case $p = 0.5$ ($k = 2$).
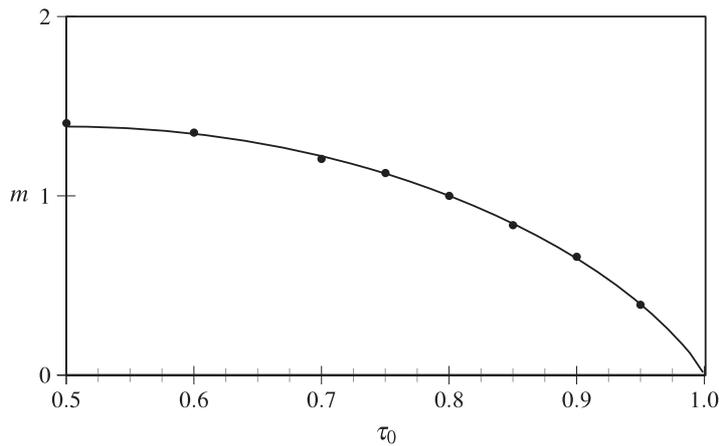

Figure 7. Slope m of linear scaling of saturation depth versus the fraction of TRUE inputs τ_0 for the critical case, $p = 1/2$. The solid line is the prediction of equation (9), $m = -2h(\tau_0)$.

connectives of level $n+1$ for every possible number of TRUE inputs that might come from level n. For each $T_n = 0, 1, \ldots L$, we choose a set of truth values for the inputs of each connective with the correct probability, setting each one TRUE or FALSE with probability T_n/L or $1 - T_n/L$, respectively. This determines, for each possible T_n, the outputs of the connectives at level $n + 1$. However, we have not yet chosen the wiring by which these connectives' inputs correspond to connectives on the level n. The construction thus far yields a proto-circuit such as the one shown in figure 8.

It is important to note that evaluating all possible inputs for a given level does not lead to a combinatorial explosion, since there are only $L + 1$ possible values for T_n. Thus we choose just $L + 1$ instances, or 'possible worlds,' at each level, and we can do this in parallel with $\mathcal{O}(L)$ processors.
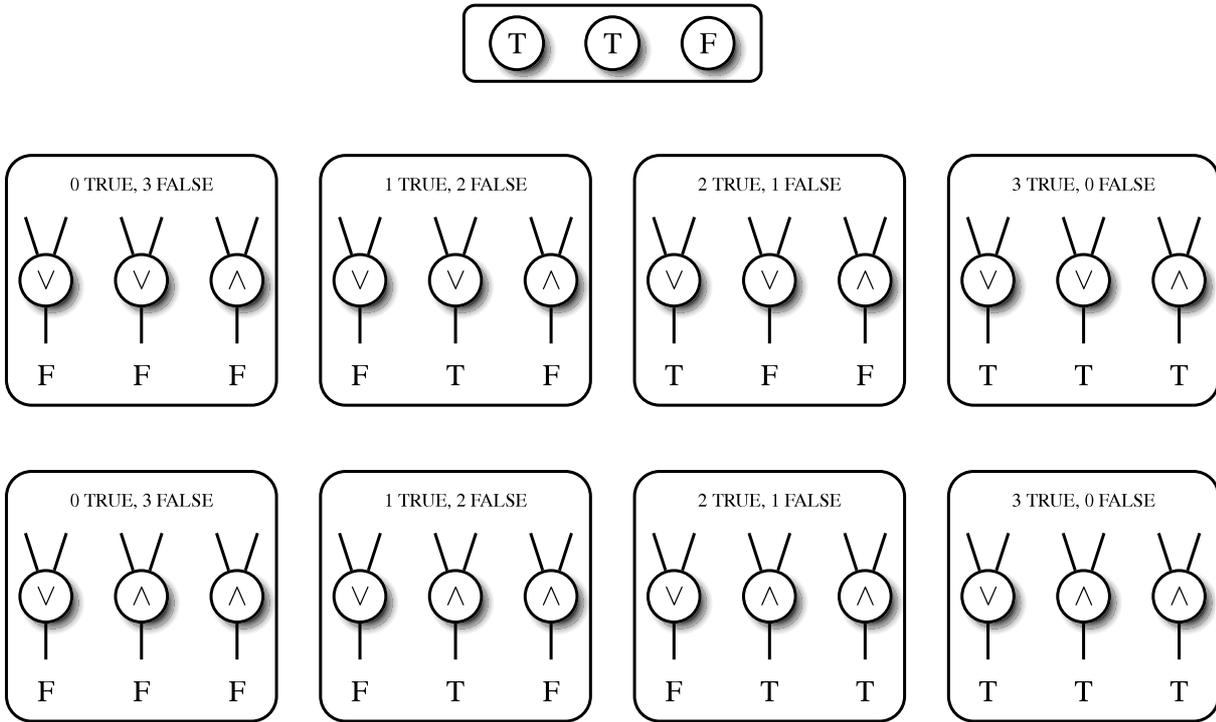
Figure 8. A proto-circuit composed of two-input AND and OR gates of width $L = 3$ and two levels. For each level and each number of TRUE inputs to a level, each gate in the level is independently evaluated.

The next step in the construction is to connect these possible worlds in a consistent way. Figure 9 shows how to do this. We create a directed graph where each vertex (n, m) corresponds to the instance of level n with m TRUE inputs. Having chosen what the outputs of the connectives will be for each value of the inputs, we draw an edge from (n, m) to the corresponding vertex $(n + 1, T_{(n,m)})$, where $T_{(n,m)}$ is the number of TRUE outputs of the instance of level n with m TRUE inputs. A logically consistent history is then the unique directed path starting at $(0, T_0)$, shown in figure 9. We can find paths through directed graphs in polylogarithmic parallel time as a function of the total number N of vertices [16]. If the circuit has width L and depth n, then $N = n(L + 1)$.

The final step in constructing the circuit is randomly connecting each connective to the ones in the previous level. Having chosen the truth values of its inputs, we simply choose each of its TRUE inputs randomly with replacement from the connectives with TRUE outputs at the previous level, and similarly for its FALSE inputs. This can be carried out independently in parallel for each connective. The result of this procedure is a properly sampled circuit and its solution, i.e. a pair (instance, solution) of random CVP, as shown in figure 10. The entire construction requires polylogarithmic parallel time on a PRAM with polynomially many processors.

We emphasize that we can carry out this construction even if the types of the connectives are not independent and identically distributed at each level. This distribution can vary from level to level, or even be highly correlated within or between levels, as long
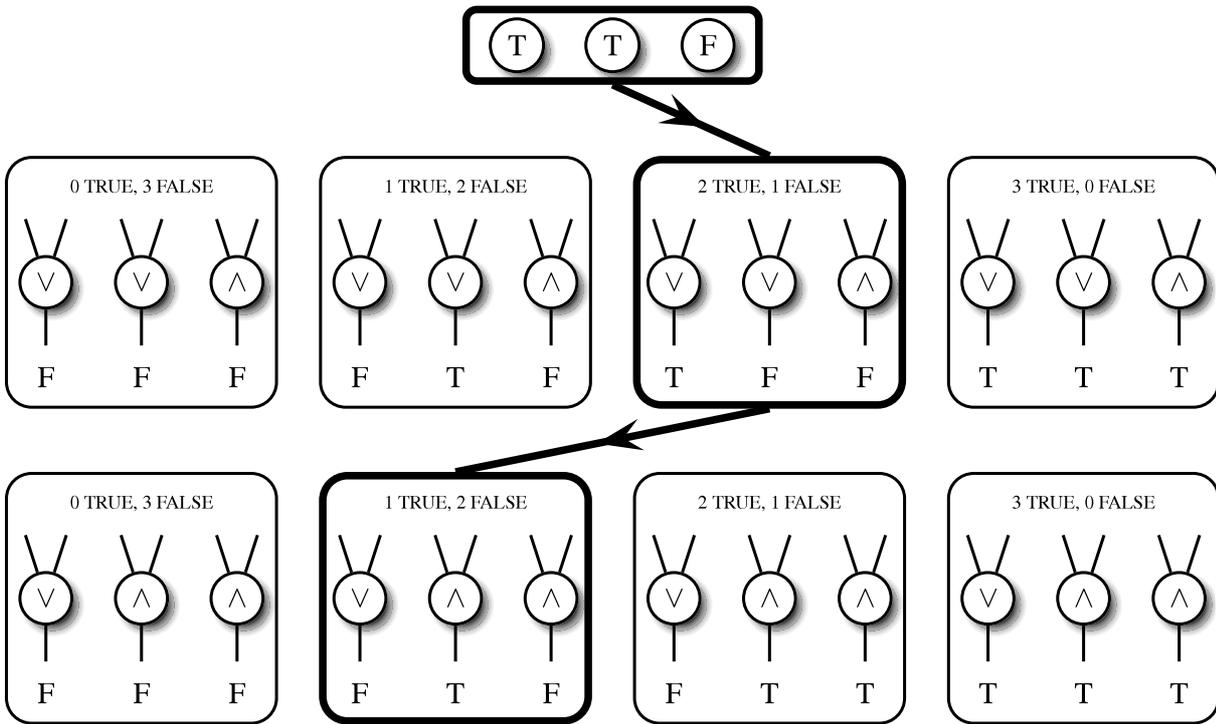
Figure 9. Pruning the proto-circuit. Instances of each level are consistently connected to the succeeding level and the subgraph connected to the circuit input is identified.
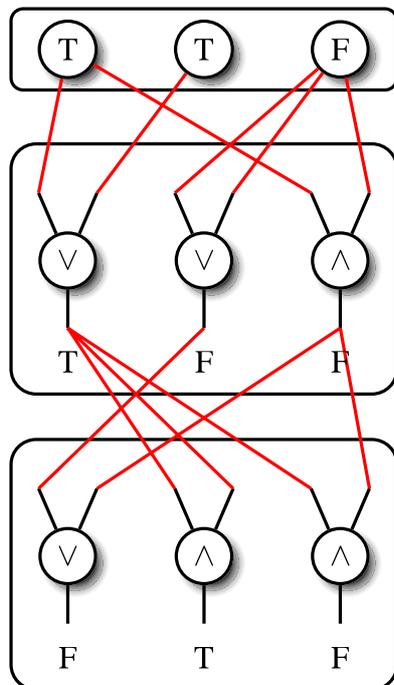

Figure 10. The circuit and its solutions is generated by randomly connecting gates at successive levels consistent with the given output of each gate.

as the joint distribution of the set of nL connective types can be sampled in **NC**: that is, as long as there is a PRAM program that runs in $\mathscr{O}(\log^k nL)$ time, for some constant k, that takes a seed with $\text{poly}(nL)$ random bits and produces a sample from the joint distribution. The only randomness we need for the construction to work is that the inputs to each connective are chosen uniformly and with replacement from those at the previous level.

8. Discussion

We have studied several random ensembles of feedforward Boolean circuits and found that, depending on the types of connectives and the fraction of connectives of each type, the circuit may be easy or hard to evaluate in parallel. The easy circuits rapidly saturate to a single truth value or, in the case of NOR circuits, a period-two oscillation between TRUE and FALSE. For these ensembles, it is only necessary to evaluate the circuit to logarithmic depth to learn its ultimate output, since nothing changes after saturation has occurred. On the other hand, for other choices of random ensembles, saturation occurs slowly, if at all, and circuit evaluation is presumably hard to carry out in parallel. Thus, although the monotone and NOR circuit value problems studied here are all **P**-complete, this worst-case classification does not distinguish among different possibilities for the average case complexity of parallel evaluation of the circuit.

When there is a single attracting fixed point, as was found for monotone circuits with more than two inputs, the exact evaluation of the circuit is to hard to accomplish in parallel but the statistical properties of the outputs at each level are predictable and insensitive to the specific inputs and wiring of the circuit. For more complicated connectives, including some with just five inputs, the recursion relations for the expected number of TRUE values on a level can lead to chaotic dynamics. For these ensembles of random circuits, even the statistical properties of the outputs of each level are hard to predict since the fraction of TRUE outputs at each level is extremely sensitive to the initial truth values and the wiring of the circuit.

In contrast to evaluating a given instance of a Boolean circuit, which may be easy or hard to accomplish in parallel, it is always easy to sample in parallel evaluated instances chosen from a random ensemble—that is, (instance, solution) pairs. The underlying idea is that the output of each connective can be chosen independently from the correct probability distribution as a function of the number of TRUE inputs in the previous level. Once the full set of possibilities is generated for each level, a consistent history is a path through a directed graph of polynomial size, and the whole construction of the circuit can be completed in polylogarithmic parallel time on a PRAM. One interesting consequence of this result, combined with the existence of chaotic connectives, is that it is possible to generate a chaotic sequence of numbers in a time that is polylogarithmic in the length of the sequence.

In statistical physics, we are often concerned with sampling ensembles of random instances of problems together with their solutions. The sampling result for random Boolean circuits holds out the promise that for other problems of interest in statistical physics, it may be possible to sample instances together with solutions with less computational effort than the traditional method of first generating an instance and then solving it.

Acknowledgments

References

[1] Monasson R, Zecchina R, Kirkpatrick S, Selman B and Troyansky L, *Determining computational complexity from characteristic 'phase transitions'*, 1999 *Nature* **400** 133
[2] Mézard M and Montanari A, 2009 *Information, Physics, and Computation* (Oxford: Oxford University Press)
[3] Moore C and Mertens S, 2011 *The Nature of Computation* (Oxford: Oxford University Press)
[4] Achlioptas D and Coja-Oghlan A, *Algorithmic barriers from phase transitions*, 2008 *49th Annual IEEE Symp. on Foundations of Computer Science* pp 793–802
[5] Krzakala F and Zdeborová L, *Hiding quiet solutions in random constraint satisfaction problems*, 2009 *Phys. Rev. Lett.* **102** 238701
[6] Greenlaw R, Hoover H J and Ruzzo W L, 1995 *Limits to Parallel Computation: P-Completeness Theory* (Oxford: Oxford University Press)
[7] Immerman N, 1999 *Descriptive Complexity* (*Graduate Texts in Computer Science*) (New York: Springer)
[8] Valiant L G, *Short monotone formulae for the majority function*, 1984 *J. Algorithms* **5** 363
[9] von Neumann J, *Probabilistic logics and the synthesis of reliable organisms from unreliable components*, 1956 *Automata Studies* ed C E Shannon and J McCarthy (Princeton, NJ: Princeton University Press) pp 43–98
[10] Moore E F and Shannon C E, *Reliable circuits using less reliable relays*, 1956 *J. Franklin Inst.* **262** 191
[11] Brodsky A and Pippenger N, *The Boolean functions computed by random Boolean formulas or how to grow the right function*, 2005 *Random Struct. Algorithms* **27** 490
[12] Mozeika A, Saad D and Raymond J, *Noisy random Boolean formulae: a statistical physics perspective*, 2010 *Phys. Rev. E* **82** 041112
[13] Díaz J, Serna M J, Spirakis P and Tsukiji T, *The average complexity of the circuit value problem*, http://www.lsi.upc.edu/~diaz/papersd/cvp.ps.gz
[14] Redner S, 2001 *A Guide to First-Passage Processes* (Cambridge: Cambridge University Press)
[15] Strogatz Steven H, 1994 *Nonlinear Dynamics And Chaos: With Applications To Physics, Biology, Chemistry, And Engineering* (*Studies in Nonlinearity*) (Cambridge, MA: Perseus Books Group)
[16] Gibbons A and Rytter W, 1988 *Efficient Parallel Algorithms* (Cambridge: Cambridge University Press)