Journal of Statistical Physics, Vol. 70, Nos. 3/4, 1993

The Computational Complexity of Pattern Formation

Jonathan Machta[1]

Received March 24, 1992; final June 8, 1992

The computational complexity of diffusion-limited aggregation and fluid invasion in porous media is studied. The time requirements on an idealized parallel computer for simulating the patterns formed by these models are investigated. It is shown that these growth models are P-complete. These results provide strong evidence that such pattern formation processes are inherently sequential and cannot be simulated efficiently in parallel.

KEY WORDS: Pattern formation; computational complexity; DLA; fluid invasion; parallel computers; P-completeness.

1. INTRODUCTION

Complex structures formed by a variety of equilibrium and nonequilibrium processes have been the subject of intensive investigation in recent years. Simple nonequilibrium growth rules such as those which define diffusion-limited aggregation (DLA) and fluid invasion in porous media yield complex patterns.[1] Equilibrium systems at critical points also generate intricate patterns, the simplest example perhaps being the infinite cluster at the percolation threshold. Efforts to characterize these patterns have focused mainly on their self-similar properties: critical exponents, fractal dimensions, and multifractal analyses. Here we examine pattern formation from the standpoint of computational complexity. More specifically, we shall ask the following question: How long will it take to generate a pattern of a given size on a parallel computer?

The motivations for investigating the parallel computational complexity of pattern formation are both practical and theoretical. With the increasing availability of parallel computers it is important to classify the

[1] Department of Physics and Astronomy, University of Massachusetts, Amherst, Massachusetts 01003.

problems of statistical physics according to whether they admit efficient parallelization.[2] For example, efficient algorithms for identifying percolation and Ising clusters are currently under study[(2)] as part of an effort to implement Monte Carlo simulations on parallel computers such as the Connection Machine™.

More fundamentally, we would like to understand how to characterize the physical complexity of pattern formation models. Bennett[(3)] identified an appealing criterion, the "slow growth law," which any suitable measure of complexity should satisfy. A physical system is considered complex if its states could not plausibly have arisen quickly. To formalize this notion, one must be able to simulate the system on a computer and measure the time required to reach typical states of the system. The time required by an optimal algorithm (referred to as "logical depth" in ref. 3) is then the measure of the complexity of the system. Since the physical world evolves in parallel, it seems appropriate to measure time on a parallel computer. Another argument for using parallel time arises from requiring that complexity be an intensive quantity. Simulating two independent, statistically similar systems on a sequential machine requires twice as much time as simulating a single system, whereas on a parallel computer with sufficiently many processors the time is nearly the same.

Growth models[(1)] such as DLA, fluid invasion, growing percolation, and invasion percolation generate patterns via a step-by-step process. As defined, these models are naturally implemented on sequential computers. For a system of size N they require polynomial time, $O(N^k)$ for some k, to reach their final states. The pattern forming process seems to be history dependent and would appear to satisfy a slow growth law. On the other hand, it may be that a DLA pattern or percolation cluster could be generated by a cleverly designed parallel algorithm that runs in polylog time, $O(\log^k N)$ for some k, using only a polynomial number of processors. The main result of the present paper is that DLA is inherently sequential and there is almost certainly no parallel algorithm which can generate the resulting patterns in polylog time. By contrast, percolation clusters can be generated in polylog time using the parallel algorithm for connected components of a graph.[(4)]

There is a well-developed theory of parallel time complexity in the computer science literature.[(4–6)] Within this theory, the idea that a problem is inherently sequential and cannot be solved on an idealized parallel computer in polylog time is formalized by the concept of P-completeness.

[2] By "efficient" we mean a parallelization in which many processors significantly speed up the construction of a single realization of a system. We do not consider the simple approach in which each processor works on a different realization of a system.

More precisely, then, the central result of this paper is that decision problems based on DLA and fluid invasion are P-complete.

We close this section with a brief overview of previous research connecting statistical physics and computational complexity. Substantial progress has been made in understanding the computational complexity of various problems in classical equilibrium statistical mechanics.(7, 8) Computing the partition function of the Ising model on a general graph is #P-complete (the enumeration counterpart of NP-complete(9)), which strongly suggests that it cannot be done in polynomial time. On the other hand, provably good polynomial time approximations can be obtained using Monte Carlo methods.(7) Finding the ground-state energies for spin glasses in more than two dimensions(10) and for self-avoiding walks on random lattices in more than one dimension(11) are NP-complete problems. On the other hand, the ground-state energy of the directed self-avoiding walk(12) and the random-field Ising model(13) are polynomial time problems. The present paper is the first to consider nonequilibrium pattern formation problems from the standpoint of computational complexity.

Cellular automata models have been extensively studied from the standpoint of simulating physical systems in parallel.(14) Because they are spatially extended and locally connected, cellular automata are attractive models of computation from the standpoint of statistical physics and have been proposed as the proper context for defining physical complexity as logical depth.(3) Some cellular automata rules (e.g., the game of life) can carry out universal computations, which implies that simulating these cellular automata are P-complete problems.(6) Cellular automata of fixed dimensionality are weaker than the idealized parallel computers, called P-RAMs, invoked in parallel complexity theory. Any problem which involves the interaction of M data bits will require time $O(M^{1/d})$ on a d-dimensional cellular automaton because of the time taken to propagate information across the system, whereas many such problems can be solved in polylog time on a P-RAM. On the other hand, the conventional definition of "time" for P-RAMs is unphysical because no allowance is made for the time required for signals to propagate between processors and memory locations.

The paper is organized as follows. Section 2 is an introduction to the theory of parallel computational complexity and P-completeness. The fluid invasion model is defined in Section 3, and in Section 4 a decision problem based on this model is proved to be P-complete. In Section 5 and an accompanying appendix, a restricted version of the model isomorphic to DLA is shown to be P-complete. The paper closes with a discussion.

2. INTRODUCTION TO THE THEORY OF PARALLEL COMPUTATIONAL COMPLEXITY

In the theory of parallel computational complexity[4-6] there are two main classes of decision problems: P and NC. Informal descriptions of these classes are provided in what follows. P is the class of problems that can be solved in polynomial time on a sequential computer such as a Turing machine or random access machine (RAM). NC is the subset of P consisting of problems solvable in polylog time on a parallel computer with a number of processors that grows as a polynomial in the problem size. The restriction to a polynomial number of processors is required since any problem can be solved in polylog time given exponentially many processors. In any case, exponentially many processors is infeasible except for very small problem sizes.

P-complete problems are those problems in P which are hardest to solve in parallel. The notion of P-completeness is formalized by considering reductions from one problem to another. Roughly speaking, problem R is NC-reducible to problem S, written $R \leqslant S$, if there exists a parallel algorithm for R which is allowed to look up solutions to S and solves R in polylog time with polynomially many processors. Thus, if $S \in NC$ and $R \leqslant S$, then $R \in NC$. A problem S is P-complete if $S \in P$ and if for all $R \in P$, $R \leqslant S$. Note that if any P-complete problem is in NC, then $P = NC$.

The identification of a first P-complete problem requires proving that a reduction exists from every problem in P to the given problem. Once a collection of P-complete problems is available, an additional problem R can be shown to be P-complete by the simpler procedure of proving that $S \leqslant R$ where S is already known to be P-complete.

The canonical P-complete problem is the CIRCUIT VALUE PROBLEM (CVP). The problem is to compute the truth value of a Boolean expression on a given set of inputs. A topologically ordered Boolean circuit of size M is a sequence, $B = (B_1,..., B_M)$. For each i, B_i is either TRUE, FALSE, or a Boolean statement of the form $B_i = (B_{i_1} \vee B_{i_2})$, $B_i = (B_{i_1} \wedge B_{i_2})$, or $B_i = \sim B_{i_1}$, where $i_1, i_2 < i$. An instance of CVP is a topologically ordered Boolean circuit. The solution of CVP is the truth value of B_M. Ladner showed that CVP is P-complete.[4, 15] The proofs presented in Sections 4 and 5 consist of reductions from variants of CVP to the fluid invasion problem.

All P-complete problems are equally difficult in the sense that if one could be solved in polylog time, then all problems in P could be solved in polylog time and P would equal NC. It is strongly believed, however, that $NC \neq P$, which implies that no P-complete problem can be solved in polylog time with polynomially many processors. At present, there is no proof that $\mathrm{NC} \neq \mathrm{P}$ and the strongest evidence in favor of this conjecture is the absence of a fast parallel algorithm for any P-complete problem. The P-completeness results obtained here for fluid invasion and DLA are thus strongly suggestive that these pattern formation processes require polynomial time on any parallel computer.

The situation is analogous to the more familiar theory of NP-completeness.[9] NP-complete problems are the hardest problems in NP and are widely believed to require exponential time to solve. If $\mathrm{NP} \neq \mathrm{P}$, then no NP-complete problem can be solved in polynomial time. Although there is strong evidence to suggest that $\mathrm{NP} \neq \mathrm{P}$, no proof has been found. Settling either of the conjectures, $\mathrm{P} \neq \mathrm{NC}$ or $\mathrm{P} \neq \mathrm{NP}$, would constitute a major breakthrough in computer science, but is likely to be exceedingly difficult.

In this paragraph several technical issues are briefly discussed. First, there are a variety of models of parallel computation. The idealized model envisioned in the theory of P-completeness has many processors working synchronously, each connected to a central controller and a shared random access memory. This is the P-RAM model of computation. In defining time on a P-RAM it is supposed that memory access can be accomplished in a number of clock cycles that is independent of the number of processors or memory locations. Variants of the model which resolve read and write conflicts between processors in different ways[4] are not relevant to defining NC and P-completeness. Second, the notion of reducibility can be defined[5] in a variety of ways and NC-reducibility is often replaced by the notion of log-space reducibility. Again, these distinctions are not central to the present discussion. Finally, complexity classes are defined for *decision* problems, e.g., problems that have yes or no answers. Our interest is to find the patterns formed by growth models, but this task can be posed as N decision problems corresponding to whether the N sites of the lattice are occupied or not by the invading fluid or DLA cluster. If the decision problem for an arbitrary site could be shown to be in NC, the whole pattern could be obtained in polylog time by increasing the number of processors by a factor of N and solving the decision problems for all lattice sites in parallel.

3. THE FLUID INVASION MODEL

The general growth model considered here is abstracted from the physics of fluid invasion in porous materials in the limit of large flow speed where surface tension can be neglected. The physical setting is a porous material filled with a viscous fluid such as oil. A much less viscous fluid such as air is injected into the system at a source and pushes the more viscous fluid out of the porous material through a sink until a path of the invading fluid connects the source and sink. If there is little or no randomness in the porous material, then the invading fluid develops viscous fingers. If the randomness is stronger, fractal patterns resembling DLA clusters develop.[16] As a first step in understanding this problem from the point of view of computational complexity, we consider a more general situation where the fluid flows on an arbitrary graph. Following Chan *et al.*,[17] we lump the local permeabilities of the material into edge conductances and the local porosities into vertex capacities.

Consider then the following general pattern formation problem, which we call FLUID INVASION:

INSTANCE. A graph $G(V, E)$. Nonnegative real "conductances" K_{ij} are assigned to each edge $\{i, j\} \in E$ and nonnegative real "capacities" ϕ_i are assigned to each vertex $i \in V$. There are two distinguished vertices s and s' which serve as a source and sink, respectively.

PROBLEM. Find the set of vertices C^* filled by the invading fluid according to the fluid invasion model, defined as follows[18]:

As a function of time t, the volume of viscous fluid at each vertex is $\sigma_i(t)$ and the pressure at each vertex is $p_i(t)$. At every time the set of vertices is divided into three disjoint subsets,

$$\begin{aligned} C(t) &= \{i \in V \mid \sigma_i(t) = 0\} \\ \partial C(t) &= \{i \in V \mid i \notin C(t) \text{ and } \exists j \in C(t)\ \{i, j\} \in E\} \\ F(t) &= V - C(t) - \partial C(t) \end{aligned} \tag{1}$$

$C(t)$ is the cluster of pores already completely filled with the invading fluid at time t. $\partial C(t)$ is the perimeter, consisting of sites which still contain the viscous fluid but which are adjacent to pores filled with the invading fluid. $F(t)$ is the collection of pores still completely filled with viscous fluid and not adjacent to a member of the cluster. The pressure in the invading fluid is unity and the pressure is zero at the sink. The pressure in the viscous fluid satisfies a discrete, inhomogeneous Laplace equation,

$$\begin{aligned} i \in C(t) \cup \partial C(t) &\rightarrow p_i(t) = 1 \\ p_{s'}(t) &= 0 \\ i \in F(t) &\rightarrow p_i(t) = \sum_j p_j(t) K_{ij} \Big/ \sum_j K_{ji} \end{aligned} \tag{2}$$

The evolution of the fluid configuration is governed by Darcy's law,

$$\dot{\sigma}_i(t) = -\sum_j K_{ij}[p_i(t) - p_j(t)] \tag{3}$$

with initial condition that the cluster is the source vertex and every other vertex is filled with the viscous fluid,

$$C(0) = \{s\} \qquad \text{and for all} \quad i \in E - \{s\}, \quad \sigma_i(0) = \phi_i \tag{4}$$

The cluster grows until there is a path from s to s', i.e., $s' \in \partial C$. The completed cluster is denoted as C^*. The growth of the cluster can be simulated by an iterative algorithm with discrete time steps:

FLUID INVASION ALGORITHM. Suppose that some vertex i_n has just joined the cluster at time t_n [i.e., $\sigma_{i_n}(t_n) = 0$ but $\sigma_{i_n}(t) > 0$ for $t < t_n$]. The pressures are computed at each vertex in $F(t_n)$ by solving the set of linear equations (2). The emptying rate $\dot{\sigma}_i(t_n)$ of each perimeter vertex is obtained from (3) and at time t_{n+1} the next vertex i_{n+1} joins the cluster, where t_{n+1} and i_{n+1} are obtained from

$$t_{n+1} = t_n + \min_{i \in \partial C(t_n)} \{\sigma_i(t_n)/\dot{\sigma}_i(t_n)\} \tag{5}$$

The remaining perimeter site capacities are updated using (3) and the neighbors of i_{n+1} are added to the perimeter. The procedure is initialized by Eq. (4) and is iterated until $s' \in \partial C$.

The fluid invasion model can also be interpreted as an electrical breakdown model. In this picture, the graph is an electrical network and K_{ij} represent bond conductances before breakdown. A perimeter site j breaks down when the net charge flowing into it directly from the cluster exceeds ϕ_j. After breaking down, the conductances to adjacent sites in the cluster become infinite.

4. COMPUTATIONAL COMPLEXITY OF FLUID INVASION

In this section we show that the FLUID INVASION is P-complete by exhibiting a reduction from the CIRCUIT VALUE PROBLEM. It is straightforward to see that CVP is P-complete when the only gate type allowed is NOR, $B_i = \sim(B_{i_1} \vee B_{i_2})$, and the circuit is topologically ordered. This version of CVP is called NOR CVP and is used in the following proof.

Theorem 1. FLUID INVASION is P-complete.

Proof. It is clear that FLUID INVASION is in P. The steps in the fluid invasion algorithm must be iterated at most N times, where N is the size of the graph. At each iteration the most complex step is the inversion of an $N \times N$ matrix to compute the pressure field according to (2).

NOR CVP is reduced to FLUID INVASION by the following construction. The idea is to show that an evaluation of a Boolean expression can be simulated by the growing cluster of a specially prepared instance of FLUID INVASION. Each NOR gate B_k is represented by a linear path from s to s' consisting of four vertices and five edges as shown in Fig. 1. The vertex $y(k)$ adjacent to s represents the output of gate k. The vertices $a(k)$ and $b(k)$ function as time delays. The third vertex $x(k)$ represents the conjunction of the inputs of gate k. The connections between gates are represented by edges. The input TRUE is represented by an edge from s to the input vertex $x(k)$, while the absence of such an edge represents FALSE. The output vertex of the last gate $y(M)$ is connected directly to the sink s'. An instance of NOR CVP and its equivalent FLUID INVASION graph are shown in Fig. 2. Each output vertex is assigned a capacity equal to the number of the gate it represents, $\phi_{y(k)} = k$. All input vertices are assigned small capacities, $\phi_{x(k)} = 1/M^5$. All of the time delay vertices are assigned large capacities, $\phi_{a(k)} = \phi_{b(k)} = M^2$. The five edges within a gate each has conductance four, e.g., $K_{s,a(k)} = 4$. The edges which connect gates to one another and to the source and sink are assigned a small conductance, $1/M^3$.

The following lemma shows how the pattern formed by the above network simulates the instance of NOR CVP.

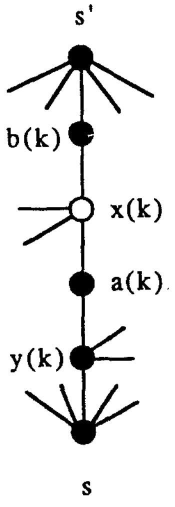

Fig. 1. The subgraph representing NOR gate k. Here $x(k)$ (open circle) is the input vertex and $y(k)$ the output vertex. $a(k)$ and $b(k)$ serve as time delays. Each edge along the path from s to s' has conductance 4.

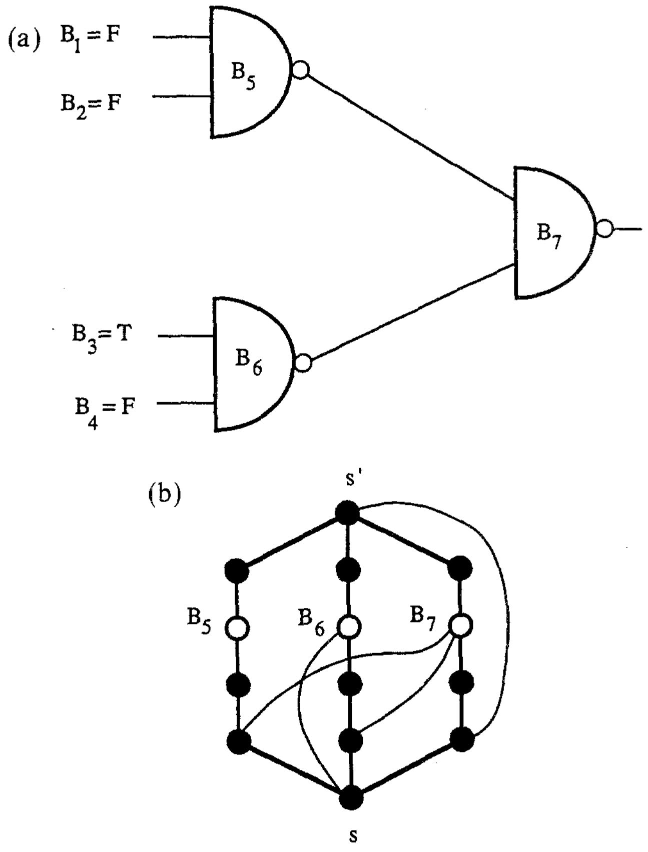

Fig. 2. (a) A Boolean circuit with four inputs and three NOR gates. (b) The corresponding FLUID INVASION graph. The bold lines represent edges with large conductance and the light curved lines represent small-conductance edges which represent connections between the gates. The inputs of vertices B_5, B_6, and B_7 are represented by open circles and have small capacities.

Lemma. $y(k) \in C^*$ if and only if $B_k = \text{TRUE}$.

Proof. Consider a given gate k. Since $y(k)$ is initially part of the perimeter ∂C, it begins to empty. While the input $x(k)$ is not part of C or ∂C, then the most important flow path by which $y(k)$ empties is the linear path through gate k that has a total conductance of one. Thus $\dot{\sigma}_{y(k)} = -1 - O(1/M)$, where the small correction is due to the parallel paths from $y(k)$ to the sink through other gates. Thus if the input $x(k)$ does not join the cluster first, then $y(k)$ will join C at time approximately k. On the other hand, suppose that $x(k)$ joins the cluster before time k. Then the path by which $y(k)$ empties is via the low-conductance connections to gates with higher indices. Since there are less than M of these, the rate at which $y(k)$ empties is now $O(1/M^2)$. Thus $y(k)$ cannot empty before a time M^2 which is always longer than the time for the cluster to reach completion, so that $y(k) \notin C^*$.

To summarize, if $x(k)$ is in the cluster, then $y(k)$ will not join the cluster, but if $x(k)$ has not joined the cluster before time k, then $y(k)$ will join the cluster approximately at time k. The gate carries out the Boolean function of negation if $y(k) \in C^*$ is interpreted as TRUE.

Next consider the effect of the state of gate k on gate m, where k and m are connected and $m > k$. First suppose that B_k has the value TRUE so that $y(k)$ becomes a member of C at time k. Then the input $x(m)$ becomes a perimeter vertex and, due to its small capacity, joins the cluster after a time less than $O(1/M)$. Thus, if either of the inputs to gate m registers TRUE before time m, then $x(m)$ joins the cluster shortly thereafter and $y(m) \notin C^*$. On the other hand, if both inputs to gate m are FALSE, then $x(m)$ is not part of the perimeter and $y(m)$ joins the cluster approximately at time m. By time m, gate m has computed the Boolean function NOR.

It is easy to check that the time delay vertices $a(k)$ and $b(k)$ ensure that gate k remains in the same logical state until time $O(M^2)$ when the cluster is completed. This completes the proof of the lemma. ▮

Since the construction of the required graph for FLUID INVASION is locally defined and requires less than $4M$ vertices, it is easy to see that the reduction is in NC. ▮

5. DLA IN TWO DIMENSIONS

Diffusion-limited aggregation[19] (DLA) is one of the most widely studied pattern formation models. In DLA the cluster grows by accretion starting from an initial "seed." Successive particles start at a source and random walk until they reach the growing cluster, where they stick. A new particle begins its random walk as soon as the previous particle is incorporated into the cluster. The cluster grows until it reaches the source or a predetermined size.

DLA clusters grow by stochastic dynamics on a uniform lattice, whereas the fluid invasion model is governed by deterministic dynamics on a random lattice. However, Chan *et al.*[17] and Koza[18] have shown that the measure associated with DLA clusters is the same as the measure associated with the fluid invasion model with a particular distribution for the quenched randomness. The fluid invasion model yields the DLA measure if the edge conductances are all the same and the vertex capacities are independent, identically distributed random variables chosen from an exponential distribution. The source and sink vertices in FLUID INVASION correspond respectively to the seed and the source for DLA.

Thus, in order to investigate the computational complexity of DLA we must consider a restricted version of FLUID INVASION in which the conductances are all equal. The most interesting model from a physical standpoint is defined on a Euclidean lattice. The restriction to a Euclidean lattice with equal conductances makes the proof of P-completeness considerably more difficult.

Theorem 2. Let $G(V, E)$ be an $L \times L$ two-dimensional square lattice together with sites s and s'. Let s be connected to all the sites on one face of the lattice and let s' be connected to all the sites on the opposite face. Let all of the bond conductances equal unity. FLUID INVASION with these restrictions is P-complete.

Proof Sketch. The proof is similar to that of Theorem 1 except that the reduction is from a restricted version of CVP in which the circuit is planar and consists only of NOT and OR gates. Goldschlager(20) proved this version of CVP P-complete. For the restricted problem, the circuit can be laid out in levels with connections only between adjacent levels.

The simulation of the Boolean circuit is again via the sequence of sites added to the growing cluster. The basic ingredient out of which the circuit components are built is a "wire" which is a connected sequence of sites having very small capacities in a background of sites having very large capacities. An example of a single wire on a 5×5 lattice is shown in Fig. 3, with the small-capacity sites shown as open circles and the background of large-capacity sites shown as filled circles. In this example, the growing cluster quickly follows the wire from s to its termination, but then no growth occurs for a very long time.

The physical mechanism which permits the simulation of Boolean functions is screening. This mechanism is illustrated by the simple NOT gate shown in Fig. 4. This gate is composed of an input wire X, an output wire Y, a power-in wire P, and a single site x, represented by a filled circle. Site x has a capacity of order unity (much larger than the wire capacities and much smaller than the insulator capacities). The cluster grows from the bottom of the diagram. The background of large-capacity, "insulating" sites (not shown explicitly) prevents the cluster from growing except along the wires. The gate is activated by the arrival of the cluster along the power-in wire, which is timed so that the input arrives before the gate is activated. Inputs and outputs are interpreted as TRUE if they are part of the completed cluster.

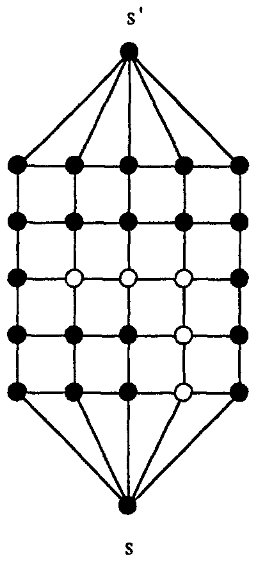

Fig. 3. A 5×5 lattice with a wire of connected very small-capacity sites (represented by open circles) in a background of very-large capacity sites (represented by solid circles). The cluster quickly grows along the wire, but then does not change for a very long time.

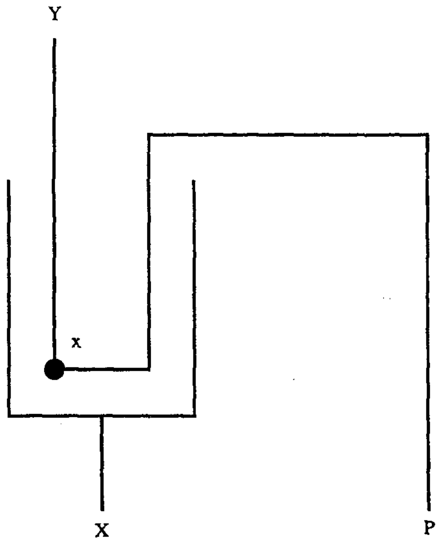

Fig. 4. A simple NOT gate. Wires composed of very-small capacity sites are indicated by bold lines. The background of very large-capacity sites is not shown, but all wires are insulated from one another by these sites. X is the input wire, P the power-in wire, and Y the output wire. The growth of the cluster is from the bottom to the top of the page.

Suppose the input to a NOT gate is FALSE. In this case, when the cluster grows along the power-in wire P, it pauses for a time of order unity at x and then progresses along the output wire Y, which then registers TRUE. Next suppose that the input is TRUE; in this case, the cluster creates a "fjord" of pressure 1 sites around the site x which screen this site. Consider what happens when the gate is activated by the arrival of the cluster at x along the wire P. In order for x to join the cluster, a flux of order unity must flow out of x along the length of the fjord; however, the pressure gradient along the fjord is very small. Thus, the cluster is effectively terminated at x and the output wire Y registers FALSE.

A quantitative discussion of the screening effect is given in the Appendix, where it is seen that the current out of x decreases exponentially with the length of the fjord. Thus, to construct a NOT gate which is stable for a time T requires a fjord whose length scales as $\log(T)$.

The NOT gate described above is the basic building block of the required fluid invasion device. In order to wire together the full circuit it is necessary to ensure that all wires are laid out on the plane without crossings and that the input signals arrive prior to the power signal at every gate. A more elaborate NOT gate meeting these requirements is shown in Fig. 5a. Note that there is now an additional power-out wire P', which activates the next gate in the network. The sites b, c, d, and e are needed to control the timing of the gates. The path of the cluster growth in the NOT gate for inputs TRUE and FALSE are shown, respectively, in Figs. 5b and 5c. Note that the extra fjords to the left of the simple NOT circuit ensure that there is no back growth of the cluster to the output wire Y.

The time delays are set up so that the input signal to the nth gate of level m arrives at time approximately $n + m\tau$ and the power signal arrives roughly at time $n + m\tau + \delta$. The time delays of sites b, x, c, d, and e are chosen to be n, $1/2$, $\tau - \delta - n - 1/2$, $\delta + 1$, and $1/2$, respectively. The input arrives at time $m\tau$, the power-in at time $m\tau + n + \delta$, and the power-out emerges at time $m\tau + n + \delta + 1$ and the output emerges at time $(m+1)\tau$. Small errors in time delays will not lead to fatal errors in the computation of the circuit value; however, a full proof would require showing that computing the site capacities needed to produce the above time delays with sufficient accuracy could be done locally and in polylog time. A crucial observation here is that the current flows in a given gate are perturbed by the status of other gates by an amount which falls off as the inverse distance to other gates, so that by increasing the separation between gates polynomially it is possible to achieve effective independence between the timing in each gate.

NOR and OR gates are easily made from the above NOT gate.

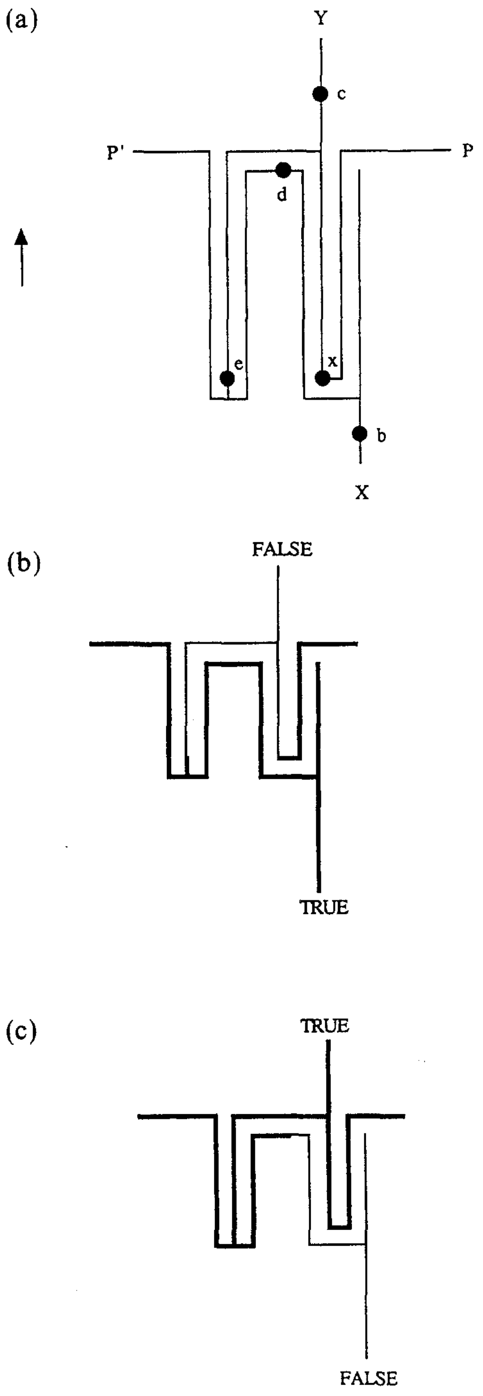

Fig. 5. (a) NOT gate. X is the input, Y the output, P the power-in, and P' the power-out wires. (b) Cluster configuration after activation of the gate with TRUE input. (c) Cluster configuration after activation of the gate if the input is FALSE. Sites in the cluster are joined by bold lines in (b) and (c).

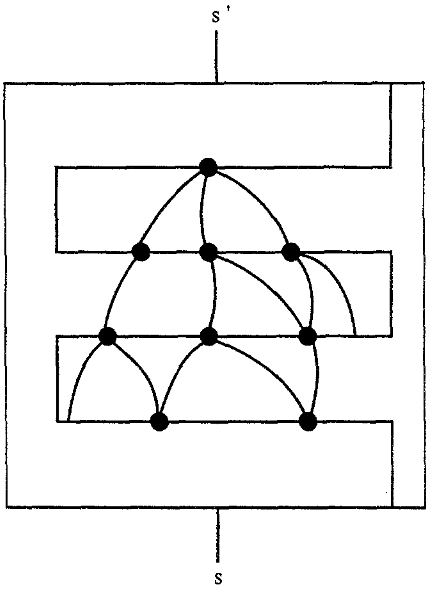

Fig. 6. The large-scale layout of a Boolean circuit as a two-dimensional FLUID INVASION problem. The bold lines indicate wires and the filled circles either OR or NOT gates (see Fig. 5 for details). The power wire, shown as straight, laces back and forth between gates at successive levels. The connections between gates at successive levels are shown as curved wires.

Joining two wires before they enter a NOT gate produces a NOR gate. Placing a simple NOT gate at the output of a NOR gate with power supplied by a tap from the main power wire produces an OR gate.

The overall structure of the circuit is shown in Fig. 6. Power wires (shown as straight) loop back and forth transversely across each level in the circuit, while wires carrying truth values (shown as curved) connect gates on adjacent levels. TRUE inputs required in level m are obtained from the power wire at level $m-1$. The large scale layout of the circuit is similar to the construction required in the proof that the lexicographically first ordered maximal path problem is P-complete.(21)

We have shown that the devices required to emulate NOT and OR gates perform correctly. The layout of the circuit can be carried out locally on a lattice whose size scales as a power of the size of the original Boolean circuit. Thus we have exhibited an NC-reduction from the planar, layered CIRCUIT VALUE problem to the two-dimensional FLUID INVASION problem with constant conductances. ∎

6. DISCUSSION.

What is the meaning of the result that FLUID INVASION and DLA are P-complete? Suppose, as is widely believed, that $NC \neq P$; then Theorems 1 and 2 imply the *existence* of instances of FLUID INVASION which cannot be solved by any parallel computer in polylog time using a polynomial number of processors. Nonetheless, it is possible that some or even most instances could be solved in fast parallel time. Thus it is also important to characterize the typical or average parallel time to solve a problem. This kind of question can be posed in a natural way within statistical physics since problem instances are equipped with measures. For example, it would be interesting to determine the average case complexity of FLUID INVASION equipped with the DLA measure (i.e., the site capacities are independent, identically distributed variables chosen from an exponential distribution). While a theory of average case complexity for NP problems is reasonably well developed,(22, 23) an equivalent theory of average case parallel complexity is not yet available.

This paper represents a first effort to bring to bear the theory of computational complexity to problems in nonequilibrium statistical physics. Understanding the computational complexity of problems in statistical physics has practical significance for numerical simulations and may also yield fundamental insights into the nature of complex physical systems.

APPENDIX

In this appendix we show that the current flowing out of a site at the end of a fjord diminishes exponentially in the length of the fjord. The equivalent circuit modeling a fjord of length N is shown in Fig. 7. All resistors have the value unity. In order to analyze this circuit, let x_j be the voltage at site j in the fjord. The node voltage equations (2) take the form

$$x_j = (2 + x_{j-1} + x_{j+1})/4 \tag{A.1}$$

with boundary conditions $x_0 = 1$ and $x_N = 0$. The general solution to (A.1) is

$$x_j = ae^{+\lambda j} + be^{-\lambda j} + c \tag{A.2}$$

Plugging this solution into (A.1) and equating terms which behave as $e^{-\lambda j}$ and $e^{+\lambda j}$ yields $c = 1$ and $\lambda = \cosh^{-1}(2) = 1.317$. Invoking the boundary conditions, one obtains the specific solution

$$x_j = 1 - \frac{\sinh(\lambda j)}{\sinh(\lambda N)} \tag{A.3}$$

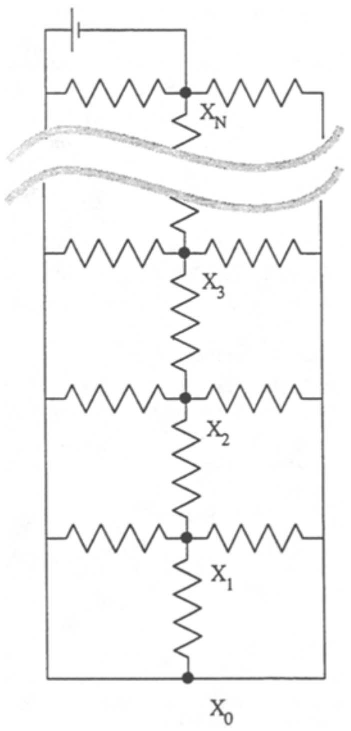

Fig. 7. Equivalent circuit representing a fjord of depth N. All resistors have unit value and the voltage drop across the battery is unity.

Thus, the current J flowing out of site 0 is, from (3),

$$J = \frac{\sinh(\lambda)}{\sinh(\lambda N)} \tag{A.4}$$

which behaves as $J \sim e^{-\lambda N}$ for large N.

ACKNOWLEDGMENTS

I am grateful to Ray Greenlaw for helpful discussions and numerous useful comments on the manuscript. I thank David Barrington, Neil Immerman, and Charles Bennett for helpful discussions. This work was supported in part by NSF grant DMR 9014366.

REFERENCES

1. T. Vicsek, *Fractal Growth Phenomena* (World Scientific, Singapore, 1989).
2. J. Apostolakis, P. Coddington, and E. Marinari, *Europhys. Lett.* **17**:189 (1992); R. C. Brower, P. Tamayo, and B. York, *J. Stat. Phys.* **63**:73 (1991).
3. C. H. Bennett, in *Complexity, Entropy, and the Physics of Information*, Wojciech H. Zurek, ed. (Addison-Wesley, 1990).
4. A. Gibbons and W. Rytter, *Efficient Parallel Algorithms* (Cambridge University Press, Cambridge, 1988).
5. S. A. Cook, *Information Control* **64**:2 (1985).
6. R. Greenlaw, H. J. Hoover, and W. L. Ruzzo, A Compendium of Problems Complete for P, Department of Computer Science, University of New Hampshire, Technical Report TR 91-14 (1991).
7. M. Jerrum and A. Sinclair, *SIAM J. Computing*, to appear.
8. R. Ladner, *SIGACT News* **7**:18 (1975).
9. M. R. Garey and D. S. Johnson, *Computers and Intractability* (Freeman, San Francisco, 1979).
10. F. Barahona, *J. Phys. A: Math. Gen.* **15**:3241 (1982).
11. J. Machta, *J. Phys. A: Math. Gen.* **25**:521 (1992).
12. D. A. Huse and C. L. Henley, *Phys. Rev. Lett.* **54**:2708 (1985).
13. F. Barahona, *J. Phys. A: Math. Gen.* **18**:L673 (1985).
14. S. Wolfram, *Theory and Applications of Cellular Automata* (World Scientific, Singapore, 1986).
15. D. J. A. Welsh, in *Disorder in Physical Systems*, G. R. Grimmett and D. J. A. Welsh, eds. (Oxford University Press, Oxford, 1990).
16. J.-D. Chen and D. Wilkinson, *Phys. Rev. Lett.* **55**:1892 (1985).
17. D. Y. C. Chan, B. D. Hughes, L. Paterson, and C. Sirakoff, *Phys. Rev. A* **38**:4106 (1988).
18. Z. Koza, *J. Phys. A: Math. Gen.* **24**:4895 (1991).
19. T. A. Witten and L. M. Sander, *Phys. Rev. Lett.* **47**:1400 (1981).
20. L. Goldschlager, *SIGACT News* **9**:25 (1977).
21. R. Anderson and E. Mayr, *Information Process. Lett.* **24**:121 (1987).
22. L. A. Levin, *SIAM J. Comput.* **15**:285 (1986).
23. Y. Gurevich, *J. Computer Syst. Sci.* **42**:346 (1991), and references therein.