Parallel dynamics and computational complexity of network growth models

Benjamin Machta[1,2] and Jonathan Machta[1]

[1]*Department of Physics, University of Massachusetts, Amherst, Massachusetts 01003-3720, USA*
[2]*Department of Physics, Brown University, Box 1843, Providence, Rhode Island 02912, USA*

The parallel computational complexity or depth of growing network models is investigated. The networks considered are generated by preferential attachment rules where the probability of attaching a new node to an existing node is given by a power α of the connectivity of the existing node. Algorithms for generating growing networks very quickly in parallel are described and studied. The sublinear and superlinear cases require distinct algorithms. As a result, there is a discontinuous transition in the parallel complexity of sampling these networks corresponding to the discontinuous structural transition at $\alpha=1$, where the networks become scale-free. For $\alpha>1$, networks can be generated in constant time while for $0 \leq \alpha < 1$, logarithmic parallel time is required. The results show that these networks have little depth and embody very little history dependence despite being defined by sequential growth rules.

I. INTRODUCTION

This paper is concerned with the complexity of networks. Many features of biological, social, and technological systems can be described in terms of networks. Examples include gene networks, friendship networks, citation networks, the power grid, the Internet, and the world wide web [1]. Although the systems that generate these networks are extremely complex, the networks themselves may or may not evidence this complexity. In many cases, the networks generated by complex systems are approximately scale-free. Barabasi and Albert [2] (BA) showed that scale-free networks can be generated by rules for network growth that embody the intuitively plausible idea of preferential attachment [18]. In their model, the network grows by the addition of one node at a time and each node creates one new connection to an existing node. Existing nodes in the network that already have many connections are more likely to gain the new connection from the new node added to the network. The growing network model seems to incorporate a history-dependent process, albeit simplified, into the generation of the network.

One of the essential markers of complexity is a long history. Complex systems cannot arise instantaneously but require a long sequence of interactions to develop. Neither "complexity" nor "long history" are well-defined concepts but an appropriate proxy for these ideas can be formulated within computational complexity theory. Computational complexity theory is concerned with the resources required to solve problems. Although there are various resources required to solve computational problems, here we focus on parallel time or depth. Depth is the number of computational steps needed by a parallel computer to solve a problem. In our case, the problem is to generate a statistically correct representation of the network. If the depth of the computation needed to generate the network is large, even using the most efficient algorithm, we say that the network has a long history and cannot be generated quickly. If, on the other hand, only a few parallel steps are needed to generate the network, then it cannot be complex.

The BA growing network model would appear to have substantial depth since nodes are added to the network one at a time and the preferential attachment rule uses knowledge of the existing state of the network to decide where each new node will attach. If the BA model captures the mechanism for the scale-free behavior found in real world networks, then perhaps one can conclude that some of the complexity or history dependence of the social, biological, or technological system that generated the network is embodied in the network. One of the main conclusions of this paper is that growing network models do not actually embody much history dependence. What we show is that there is a fast parallel algorithm that generates BA growing networks with N nodes in $\mathcal{O}(\log \log N)$ steps.

The BA model has a linear preferential attachment rule. Krapivsky, Redner, and Leyvraz [3] introduced a generalization of the BA model in which the probability to connect to a node is proportional to a power α of its number of connections. The original BA model is the case $\alpha=1$ while $\alpha=0$ is a random network. The class of models $0 \leq \alpha < \infty$ is analyzed in Refs. [3,4] and it is seen that $\alpha=1$ marks a "phase transition" between a "high-temperature phase" for $\alpha<1$ where no node has an extensive number of connections and a "low-temperature phase" for $\alpha>1$ where a single node has almost all connections in the large N limit.

We show that distinct but related parallel algorithms are needed to efficiently simulate the $\alpha<1$ and $\alpha>1$ regimes so that there is a discontinuous transition in the computational complexity of simulating the model at $\alpha=1$. For $0<\alpha<1$, the parallel time for generating a network of size N scales logarithmically in N while for $1<\alpha<\infty$ there is a constant time algorithm. Exactly at $\alpha=1$ still a third algorithm is most efficient with parallel running time that is $\mathcal{O}(\log \log N)$.

Although the primary purpose for constructing the algorithms presented here is to elucidate the complexity and history dependence of network models, the strategies embodied in the algorithms may have practical applications in computational studies of stochastic processes. For example, the redirection method that is used to show logarithmic complexity

in the $\alpha=1$ case is closely related to the best known sequential method for simulating this model [4].

A number of nonequilibrium models in statistical physics defined by sequential rules have been shown to have fast parallel dynamics. Examples include the Eden model, invasion percolation, the restricted solid-on-solid model [5], the Bak-Sneppen model [6], and internal diffusion-limited aggregation [7], all of which can be simulated in parallel in polylogarithmic time. On the other hand, no polylog time algorithm is known for generating diffusion-limited aggregation clusters and there is evidence that only power-law speedups are possible using parallelism [8,9].

Phase transitions in computational complexity have been the object of considerable recent study; for example, see Ref. [10]. Most of the attention has been focused on **NP**-hard combinatorial optimization problems. Growing networks and many other physically motivated models are naturally related to problems in the lower class **P** (problems solvable in polynomial time). One of the purposes of this paper is to provide an example of a transition in computational complexity at this lower level of the complexity hierarchy.

The paper is organized as follows. In the next section, we define and describe the class of preferential attachment network growth models to be studied. In Sec. III, we give a brief review of relevant features of parallel computational complexity theory. Section IV presents efficient parallel algorithms for sampling growing network models and related systems, Sec. V analyzes the efficiency of these algorithms, and Sec. VI presents results from numerical studies of the efficiency of one of the algorithms. The paper ends with a discussion.

II. GROWING NETWORK MODELS

In this section, we describe growing network models with preferential attachment first considered by Barabasi and Albert [2] and later generalized by Krapivsky, Redner, and Leyvraz [3,4]. Consider a graph with N ordered nodes, each having one outgoing link, constructed by the addition of one node every time step so that at time t in the construction, node t is attached to a previous node, 0 through $t-1$. The probability $\pi_n(t)$ of attaching node t to node $n<t$ is given by

$$\pi_n(t) = \frac{F(k_n(t))}{Z(t)}, \qquad (1)$$

where $k_n(t)$ is the degree (number of connections) of n, at time t, F is some function, and Z is the normalization given by

$$Z(t) = \sum_{j=0}^{t-1} F(k_j(t)). \qquad (2)$$

We require that $F(k)$ is a nondecreasing function of k. Notice that, in general, $\pi_n(t)$ is a function not only of $k_n(t)$ but also of $k_j(t)$ for all $j<t$ because of the normalization, Z. The attachment probabilities depend on all the node degrees unless $Z(t)$ is a function of t alone. This simpler form holds if and only if F is a linear function, $F(k)=a+bk$. In the latter case, $Z(t)=(a+2b)t$ since $\sum_{j=0}^{t-1} k_j(t)=2t$.

The linear homogeneous case, $F(k)=k$, corresponds to the original Barabasi-Albert model [2] and leads to a scale-free network where the degree distribution, $P(k)$, has a power-law tail, $P(k) \sim k^{-3}$. More generally, if $F(k)$ is asymptotically linear, $P(k) \sim k^\nu$, where ν is tunable to any value $2 \leqslant \nu < \infty$ [3,4,11]. The asymptotically linear attachment kernel is a marginal case and marks a "phase transition" between regimes with qualitatively different behavior. Consider the homogeneous models, $F(k)=k^\alpha$, studied in detail in Ref. [4]. In the sublinear case, $0<\alpha<1$, the degree distribution has a stretched exponential form and the node with the maximum degree has polylogarithmically many connections. The limiting case of $\alpha=0$ is a random network where each connection is randomly and independently chosen. There is an analogy between α and temperature in a thermodynamic system with the range $0 \leqslant \alpha < 1$ like a high-temperature phase. The order parameter is the maximum degree in the system divided by N and the order parameter vanishes for $0 \leqslant \alpha \leqslant 1$. In the superlinear or low-temperature phase, $\alpha>1$, there is a single "gel" node that has almost all connections and the order parameter is unity. The phase transition then has a discontinuous character despite the fact that the $\alpha=1$ state is scale-free. Another indication that the transition is discontinuous is seen by looking at the entropy. Using the Kolmogorov-Chaitin definition of entropy as the minimum number of bits required to describe a system state [12], it is clearly seen that the entropy per node is positive for all $\alpha \leqslant 1$ but that for $\alpha>1$ the entropy per node vanishes since almost all nodes connect to the gel node and it is only necessary to specify the connections for those nodes that do not connect to the gel node. Thus, the entropy per node is also discontinuous at $\alpha=1$.

III. PARALLEL COMPUTATION AND DEPTH

Computational complexity theory is concerned with the scaling of computational resources needed to solve problems as a function of the size of the problem. An introduction to the field can be found in Ref. [13]. Here we focus on parallel computation and choose the standard *parallel random access machine* (PRAM) as the model of computation [14]. The main resources of interest are *parallel time* or *depth* and number of processors. A PRAM consists of a number of simple processors (random access machines, or RAMs) all connected to a global memory. Although a RAM is typically defined with much less computational power than a real microprocessor such as Pentium, it would not change the scaling found here to think of a PRAM as being composed of many microprocessors all connected to the same random access memory. The processors run synchronously and each processor runs the same program. Processors have an integer label so that different processors follow different computational paths. The PRAM is the most powerful model of classical, digital computation. The number of processors and memory is allowed to increase *polynomially* (i.e., as an arbitrary power) in the size of the problem to be solved. Communication is nonlocal in that it is assumed that any processor can communicate with any memory cell in a single time step. Obviously, this assumption runs up against speed of

light or hardware density limitations. Nonetheless, parallel time on a PRAM quantifies a fundamental aspect of computation. Any problem that can be solved by a PRAM with H processors in parallel time T could also be solved by a single processor machine in a time W such that $W \leq HT$ since the single processor could sequentially run through the tasks that were originally assigned to the H processors. The single processor time W is sometimes referred to as the computational work. On the other hand, it is not obvious whether the work of a single processor can be reorganized so that it can be accomplished in a substantially smaller number of steps by many processors working independently during each step.

An example of where exponential speed-up can be achieved through parallelism is adding N numbers. Addition can be done by a single processor in a time that scales linearly in N. On a PRAM with $N/2$ processors, addition can be carried out in $\mathcal{O}(\log N)$ parallel time using a binary tree. For simplicity, suppose N is a power of 2. In the first step, processor one adds the first and second numbers and puts the result in memory, processor two adds the third and fourth numbers and puts the result in memory, and so on. After the first step is concluded, there are $N/2$ numbers to add and these are again summed in a pairwise fashion by $N/4$ processors. The summation is completed after $\mathcal{O}(\log N)$ steps. Addition is said to have *efficient* parallel algorithms in the sense that they can be solved in time that is a power of the logarithm of the problem size, here N, that is, *polylog* time. On the other hand, it is believed that there are some problems that can be solved in polynomial time using a single processor but cannot be efficiently parallelized. It is believed that P-complete problems [14,15] have this property and cannot be solved in polylog time with polynomially many processors.

The main concern of this paper is the complexity of generating networks defined by preferential attachment growth rules. Since these networks grow via a stochastic process, we envision a PRAM model equipped with registers containing random numbers. The essential question that we seek to answer is the depth (number of PRAM steps) required to convert a set of independent random bits into a statistically correct network.

IV. PARALLEL ALGORITHMS FOR GROWING NETWORK MODELS

At first glance, it seems that growing networks have a strong history dependence. It would appear that to connect some node t appropriately one must first connect all nodes prior to t in order to compute the connection probabilities for t according to Eq. (1). Surprisingly, one can construct a statistically correct network using an iterative parallel process that converges in far fewer than t steps. The strategy is to place progressively tighter lower bounds on the connection probabilities based on connections made in previous parallel steps in the process.

A. A coin toss with memory

A simple example of the general strategy is instructive. Consider a biased coin toss with memory such that the num-

ber of heads on the first t coin tosses modifies the probability of a head on toss $t+1$. Suppose that more heads on previous tosses increases the probability of a head on the current toss according to some function $f(x)$, where $\pi(t) = f(x(t))$ is the probability of a head on the tth coin toss and $x(t)$ is the fraction of heads on all the coins tossed before t. Suppose that f is a nondecreasing function of its argument and that $f(0) > 0$. Note that the special case $f(x) = x$ is a Polya urn problem and is discussed in Sec. IV D.

The goal is to simulate a sequence of N coin tosses. It would appear that we cannot decide coin t until we have decided all its predecessors. Nonetheless, we can proceed in parallel by successively improving lower bounds on the probability that a given coin toss is a head. Let, $p^S(t)$,

$$p^S(t) = f(x^S(t)), \tag{3}$$

be an estimated lower bound on the probability that the tth coin toss is a head on the Sth step of the algorithm where $x^S(t)$ is the fraction of tosses determined to be heads at the beginning of iteration S. The starting assumption is that none of the tosses have been determined, $x^1(t) = 0$ for all t, and this assumption is used to compute how many coins become heads on the first iteration. Thus, $p^1(t) = f(0)$ and, on the first iteration, coin t becomes a head with this probability. Once a coin becomes a head, it stays a head while coins that are not heads remain undecided. On the second iteration, we make use of the heads decided in the first iteration to recompute the fraction determined to be heads, $x^2(t)$, and from these obtain the new bounds $p^2(t) = f(x^2(t)) \geq p^1(t)$. For each coin t that is not yet determined to be a head, we declare it a head with conditional probability $\rho^2(t)$ that it will become a head on this step given that it is not yet a head, $\rho^2(t) = [p^2(t) - p^1(t)]/[1 - p^1(t)]$. Some new coins are declared heads and these are then used to compute $x^3(t)$. In general, if coin t is not yet determined by step S, it becomes a head with probability

$$\rho^S(t) = \frac{p^S(t) - p^{S-1}(t)}{1 - p^{S-1}(t)}, \tag{4}$$

where $\rho^S(t)$ is the conditional probability of coin t becoming a head on step S given it was undecided up to step S. The expression for the conditional probability follows from the observation that the denominator is the marginal probability of being undecided after step $S-1$ and the numerator is the probability of becoming a head on step S. The algorithm stops on step T when there is no change from one step to the next, $x^T(t) = x^{T-1}(t)$ for all t, and the lower bounds equal the true probabilities $p^T(t) = \pi(t)$. At the end of the simulation, every coin that is not a head is declared to be a tail. For every t,

$$x^1(t) \leq x^2(t) \leq \cdots \leq x^T(t) = x(t), \tag{5}$$

so that

$$p^1(t) \leq p^2(t) \leq \cdots \leq p^T(t) = \pi(t). \tag{6}$$

Thus the procedure is well-defined and we can decide in stages whether coin t will be a head.

In the following two sections, we show how to generalize this strategy to the case of preferential attachment growing network models.

B. Parallel algorithm for linear and sublinear kernels

This section describes a parallel algorithm for constructing a network with a sublinear or linear attachment, rule, $F(k) = k^\alpha$, where $0 \leq \alpha \leq 1$ or, more generally, the case where the attachment weight $F(k)$ is a nondecreasing, convex function of k. As in the coin toss example, on intermediate parallel steps we have nodes whose connections are not yet determined. In this algorithm, we lump all of these connections into a "ghost" node whose in-degree is equal to the number of nodes that have not yet been determined. On every parallel time step, S, the algorithm attempts to connect every node that is currently connected to the ghost node to a real node according to lower bounds on the connection probabilities determined by connections that have been made in previous steps.

In the initialization, $S=0$ step of the algorithm, a ghost node is created and all real nodes are connected to it, except node zero, which connects to itself, and node one, which also connects to node zero. Thus, for $S=0$ and every sequential time $t>1$, every real node $n<t$ has in-degree 0 and out-degree 1, except the zero node, which has both in- and out-degree equal to 1. The ghost node has in-degree $t-1$ for $t > 0$. Let $k_g^S(t)$ be the number of nodes connecting to the ghost node at the beginning of parallel step S and sequential time t so that $k_g^1(t) = t-1$ for $t>0$. In the first, $S=1$ step of the algorithm, the connection probability lower bound for node t to connect to node n, $p_n^1(t)$, is given by

$$p_n^1(t) = \begin{cases} F(2)/\widetilde{Z}^1(t), & n = 0, \\ F(1)/\widetilde{Z}^1(t), & n > 0, \end{cases} \tag{7}$$

while the connection probability, $Q^1(t)$, for the ghost node is taken to be proportional to its number of connections,

$$Q^1(t) = c(t-1)/\widetilde{Z}^1(t) \tag{8}$$

with the normalization given by

$$\widetilde{Z}^1(t) = c(t-1) + F(2) + (t-1)F(1). \tag{9}$$

The constant c is discussed below. These are the connection probabilities that would arise if each real node has one connection and the ghost node has an attachment probability proportional to its degree. On the first step of the algorithm, each node t is connected to one of its predecessors or the ghost node according to the probabilities given above.

As in the case of the coin toss model described in the previous section, on successive steps we recompute the bounds on the connection probabilities $p_n^S(t)$ for the real nodes and the ghost node $Q^S(t)$. For general S, t, and n, these probabilities are given by

$$p_n^S(t) = F(k_n^S(t))/\widetilde{Z}^S(t), \tag{10}$$

$$Q^S(t) = ck_g^S(t)/\widetilde{Z}^S(t), \tag{11}$$

with the normalization given by

$$\widetilde{Z}^S(t) = ck_g^S(t) + \sum_{m=0}^{t-1} F(k_m^S(t)). \tag{12}$$

On step S of the algorithm, the conditional probability $\rho_n^S(t)$ of connecting node t to node n, given that node t has not yet connected to a real node on an earlier step, is given by the difference between the probability bounds on successive steps divided by the marginal probability of being undetermined (connected to the ghost node) before step S,

$$\rho_n^S(t) = \frac{p_n^S(t) - p_n^{S-1}(t)}{Q^{S-1}(t)}. \tag{13}$$

Note that the denominator can be written as

$$Q^{S-1}(t) = 1 - \sum_{m=0}^{t-1} p_m^{S-1}(t). \tag{14}$$

On step S of the algorithm, each node t that was still connected to the ghost node after step $S-1$ is connected with probability $\rho_n^S(t)$ to real node $n<t$ or, with probability $\rho_g^S(t)$,

$$\rho_g^S(t) = Q^S(t)/Q^{S-1}(t) \tag{15}$$

still connected to the ghost node. The algorithm is finished after T steps when there are no more nodes connected to the ghost node and the bounds of Eq. (10) saturate to the correct probabilities of Eq. (1). Note that at least one node must connect in each parallel step since the lowest numbered node that is still unconnected will have no weight allotted to it in the ghost node.

For the conditional probabilities $\rho_n^S(t)$ to be positive, the probability bounds must be nondecreasing for all n and t,

$$p_n^1(t) \leq p_n^2(t) \leq \cdots \leq p_n^T(t) = \pi_n(t). \tag{16}$$

These inequalities imply a bound on c as follows. Since $F(k)$ is a nondecreasing function of k and $k_n^S(t)$ is a nondecreasing function of S, it is sufficient to require that $\widetilde{Z}(t)$ be a nonincreasing function, $\widetilde{Z}^S(t) \leq \widetilde{Z}^{S-1}(t)$, or, since

$$k_g^S(t) = 2t - 1 - \sum_{m=0}^{t-1} k_m^S(t), \tag{17}$$

we require that

$$\sum_{m=0}^{t-1} [F(k_m^S(t)) - ck_m^S(t)] \leq \sum_{m=0}^{t-1} [F(k_m^{S-1}(t)) - ck_m^{S-1}(t)]. \tag{18}$$

This inequality is satisfied term by term if $F(k) - ck$ is nonincreasing, which holds if

$$c \geq \max_k \{F(k+1) - F(k)\}. \tag{19}$$

Since the algorithm will finish fastest if the ghost node has the smallest possible weight, we set c equal to its lower

bound. In particular, for the power-law case, $F(k) = k^\alpha$ with $\alpha \leq 1$, the maximum occurs for $k=1$ yielding

$$c = 2^\alpha - 1. \tag{20}$$

C. The parallel algorithm for superlinear kernels

For the superlinear case, a gel node develops to which almost all nodes connect as $N \rightarrow \infty$. When $\alpha > 2$, all but a finite number of nodes connect to the gel node. The parallel algorithm described here takes advantage of the fact that the vast majority of connections are to the gel node and the gel node plays a role similar to that of the ghost node in the sublinear and linear cases. The basic structure of the algorithm is as follows. In the initialization $S=0$ phase, the sequential algorithm is run so that all nodes $t \leq t_0$ are properly connected. t_0 is chosen so that a single gel node is firmly established by the time all nodes $t \leq t_0$ are connected. The gel node is firmly established if the probability that a different node ultimately becomes the gel node is less than some small value ϵ. When α is large, t_0 is small, but as α approaches 1, for fixed ϵ, t_0 diverges. After the initialization phase, it is tentatively assumed that all nodes $t_0 < t < N$ are connected to the gel node. The gel node serves as a repository for all connections that are not yet determined. In successive steps, the connection probabilities of all nodes $t_0 < t < N$ are modified according to the number of connections that possible destination nodes, $n < t$, received in the previous step and lower bounds on connection probabilities are recalculated. The difference between old and new probability bounds is used to find conditional probabilities for moving a connection from the gel node to some other node. This process is repeated until no connections are moved from the gel node to any other node. The nodes that have not been moved away from the gel node are then determined to be connected to the gel node.

Following the general strategy, lower bounds on the connection probabilities for $t > t_0$ are determined for each parallel step,

$$p_n^S(t) = \frac{F(k_n^S(t))}{Z^S(t)}, \tag{21}$$

where the normalization is given by

$$Z^S(t) = \sum_{n=0}^{t-1} F(k_n^S(t)). \tag{22}$$

Note that the connection probabilities are calculated in the same way for the gel node and the other nodes in contrast to the sublinear case.

In the first parallel step, $S=1$, the algorithm connects every node, $t > t_0$, to some node n according to the connection probabilities $p_n^1(t)$. In successive steps, $S > 0$, it attempts to reconnect only those nodes $t > t_0$ that are still connected to the gel node. The conditional probability $\rho_n^S(t)$ for connecting $t > t_0$ to $n \neq g$ on step S is given by

$$\rho_n^S(t) = \frac{p_n^S(t) - p_n^{S-1}(t)}{p_g^{S-1}(t)}. \tag{23}$$

The numerator is the probability that t connects to n on step S and the denominator is the probability that t is undetermined after step $S-1$. The conditional probability that t is undetermined after step S, given that it was undetermined after step $S-1$, is

$$\rho_g^S(t) = \frac{p_g^S(t)}{p_g^{S-1}(t)}. \tag{24}$$

The algorithm is finished after step T if no changes occur from step $T-1$ to step T. On step T, nodes that are connected to g are considered to be determined and actually connected to the gel node.

The algorithm is valid if Eq. (16) holds for all $t > t_0$ and $n \neq g$. Since $F(k)$ is nondecreasing, we require that $Z^S(t)$ is a nonincreasing function of S. From Eq. (22) we must show that the change in Z from one parallel step to the next is either constant or decreasing for all t and S. We can write the requirement for the validity of the algorithm as

$$Z^S - Z^{S-1} = \sum_{m=0}^{t-1} [F(k_m^S) - F(k_m^{S-1})] \leq 0. \tag{25}$$

It is useful to take the gel node term out of the sum, as its behavior is different,

$$Z^S - Z^{S-1} = F(k_g^S) - F(k_g^{S-1}) + \sum_{m=0,m \neq g}^{t-1} [F(k_m^S) - F(k_m^{S-1})]. \tag{26}$$

At each parallel step, connections are switched from the gel node to other nodes. For every connection that is lost by the gel node, exactly one connection is gained by another node. We also note that because F is a concave function with a continuously increasing derivative, we can say that for any positive δk,

$$F'(k + \delta k) \delta k \geq F(k + \delta k) - F(k) \geq F'(k) \delta k. \tag{27}$$

Since k_g^S is decreasing with S and $k_{m \neq g}^S$ is increasing with S, we can rewrite Eq. (26), absorbing the contribution of the gel node into the sum to describe the entire rewiring of $k_m^S - k_m^{S-1}$ connections from the gel node to another node. We use Eq. (27) to put an upper bound on the change in size, the right-hand side (RHS) of Eq. (26),

$$Z^S - Z^{S-1} \leq \sum_{m=0,m \neq g}^{t-1} [k_m^S - k_m^{S-1}][F'(k_m^S) - F'(k_g^S)]. \tag{28}$$

The term on the RHS in the first square brackets is nonnegative. If $k_g^S > k_m^S$, then the term in the second square brackets is always negative because F' is a strictly increasing function of k. This argument shows that Eq. (25) holds and thus the algorithm is valid if the gel node remains the largest node until the end of the simulation. The value of ϵ and, thus, the choice of t_0 determines the error rate of the algorithm since the algorithm fails if and only if the gel node loses its status as having the most connections.

D. Redirection method for linear kernels and urns

This section explores the method proposed by Krapivsky and Redner [4] for the case of a linear attachment kernel. We show that this method can be used to generate the network in $\mathcal{O}(\log \log N)$ steps. The method works as follows: At sequential time t, node t is connected to any node $n < t$ with equal probability. With probability r, however, this node is redirected to the "ancestor" of n, the node that n connects to. As Krapivsky and Redner show, when $r = 0.5$, this procedure exactly reproduces the BA model [$F(k) = k$]. For other values of r, $F(k)$ is asymptotically linear and the connectivity distribution scales as $P_k \sim k^{-\nu}$, where $\nu = 1 + 1/r$. It is easy to see why redirection is equivalent to a linear kernel. A node that already has k connections has k ways to be connected from a new node since each of the k connections can serve as a redirection point for the new node. For $r = (1-r) = 1/2$, it is clear that $F(k) = kF(1)$, so this case corresponds to the homogeneous BA network.

This redirection process can be simulated in $\mathcal{O}(\log \log N)$ parallel time as follows. First, randomly connect every node to one of its predecessors. Once this is done, for every connection, with probability r, make that connection a redirectable (R) connection, otherwise make it a terminal (T) connection. All that remains is to trace every path of R connections until a T connection is reached. This can be accomplished using a standard parallel connectivity algorithm or by the following simple approach. For every node t, if its outgoing connection is type T, make no change to the connection. If its outgoing connection is type R, then it is redirected. Suppose t connects to i by an R connection and that i connects to j, then after the parallel step, t connects to j. Furthermore, if the i to j connection is type T, then the new connection from t to j is type T, otherwise it is an R connection. When all of the connections are type T, the program is done and the network is correctly wired. It is clear that this procedure requires a number of steps that scale as the logarithm of the longest chain of redirections. On average, the longest chain of redirections will behave as the logarithm of the system size. Each connection redirects with probability r. The average length of the longest chain of redirections, M, is estimated by $Nr^M \approx 1$, where N is the number of possible starting points and r^M is the probability of a chain of length M. Thus $\log N + M \log r \approx 0$ so $M \sim -\log N / \log r$. Note that the chain length saturates at $\mathcal{O}(\log N)$ rather than diverges as $r \to 1$. Even if $r \to 1$, each connection will typically halve the distance to the origin so that there are $\mathcal{O}(\log N)$ connections in the longest chain. A chain of connections of length M can be traced in $\log M$ steps, because each step will halve the length of the chain. Thus the algorithm will finish in $\mathcal{O}(\log M) = \mathcal{O}(\log \log N)$ steps.

The Polya urn model is closely related to the BA growing network model and the redirection method can be applied to efficiently sample its histories in parallel. In the simplest version of the model, an urn initially contains two balls, one red and one black. On each time step a ball is randomly selected from the urn and then replaced along with a new ball of the same color. Thus, after N steps the urn contains $N+2$ balls. The urn model has the unusual property that it

can have any limit law. For large N, the fraction of red balls approaches a constant but, with equal probability, that constant can be any value from zero to one. The limit law is thus determined by the initial choices.

The urn model can be viewed as a network where each ball is a node and the connection from one node to a predecessor represents the fact that the color of the later node was determined by the earlier node. To find the color of a given ball or node, the connections are traced back to one of the two initial balls. This representation shows that the urn model is identical to the linear network model in the limit that the redirection probability is unity. The typical longest path of connections back to the origin is $\mathcal{O}(\log N)$ since each connection will typically halve the distance to the origin. Thus the depth of sampling the history of an urn model is $\mathcal{O}(\log \log N)$.

V. EFFICIENCY OF PARALLEL ALGORITHMS

A. Efficiency of the parallel algorithm when $0 \le \alpha \le 1$

In this section we argue that for a system of size N, when $0 \le \alpha \le 1$, the parallel algorithm will finish in $\mathcal{O}(\log N)$ parallel steps and we estimate the prefactor of the logarithm. The starting point is an equation for the expected number of connections to the ghost node on the $S+1$ step given the number of connections on steps S and $S-1$,

$$\mathbf{E}(k_g^{S+1}(t)) = \sum_{t'=1}^{t-1} [k_g^S(t') - k_g^S(t'-1)]\rho_g^S(t'). \quad (29)$$

The quantity in the square brackets is 1 if t' is connected to the ghost node on step S and 0 otherwise, while $\rho_g^S(t')$, defined in Eq. (15), is the conditional probability that t' is connected to the ghost node after step S if it is connected to the ghost node before step S. Equation (29) holds for $S > 2$. Initially, $k_g^1(t) = t-1$. Specializing to the case that the attachment kernel is a pure power law with exponent α and ignoring constants that are irrelevant in the large-t limit, we have

$$\mathbf{E}(k_g^2(t)) = c/(c+1) = 1 - 2^{-\alpha}. \quad (30)$$

This result follows from the fact that the probability that node t will still be connected to the ghost node after the first step is, according to Eqs. (8) and (9), approximately $c/(c+1)$. The far RHS of the expression is obtained from Eq. (20).

To proceed further, we make two approximations. First, we ignore fluctuations and replace k_g by its average value on the RHS of Eq. (29),

$$k_g^{S+1}(t) = \sum_{t'=1}^{t-1} [k_g^S(t') - k_g^S(t'-1)] \frac{k_g^S(t')}{k_g^{S-1}(t')} \frac{\widetilde{Z}^{S-1}(t')}{\widetilde{Z}^S(t')}, \quad (31)$$

where the notation is simplified in this equation by interpreting k_g as the average number of connections to the ghost node and where Eqs. (11) and (15) have been used to expand ρ_g.

For the case of a linear attachment kernel, $c = 1$, and the normalization $\widetilde{Z}^S$ is independent of S. The ratio of normal-

izations thus drops out of the equation, and we obtain

$$k_g^{S+1}(t) = \sum_{t'=1}^{t-1} [k_g^S(t') - k_g^S(t'-1)] \frac{k_g^S(t')}{k_g^{S-1}(t')}. \tag{32}$$

For sublinear kernels, the choice of c insures that the ratio $\tilde{Z}^{S-1}(t')/\tilde{Z}^S(t')$ is less than 1 as discussed at the end of Sec. IV B. Our second approximation is to assume that this ratio is unity for the entire sublinear regime. Note that both $k_g^1(t)$ and $k_g^2(t)$ are proportional to t. It follows from Eq. (32) that $k_g^S(t)$ is proportional to t for all S and we write $k_g^S(t) = \kappa(S)t$. This substitution reduces Eq. (32) to

$$\kappa(S+1) = \frac{\kappa(S)^2}{\kappa(S-1)}. \tag{33}$$

Given our approximations, the ratio $\kappa(S)/\kappa(S-1) = 1 - 2^{-\alpha}$ for all S and the solution is $\kappa(S) = (1 - 2^{-\alpha})^S$. The estimate for the number of steps, T, needed to complete the algorithm is such that the ghost node is expected to have fewer than one node, $\kappa(T)N = 1$. This requirements leads to the result

$$T = \frac{\log(N)}{-\log(1 - 2^{-\alpha})}. \tag{34}$$

This result is compared to the numerical simulations in Sec. VI.

B. Efficiency of the parallel algorithm when $\alpha > 1$

In this section, we show that the $\alpha > 1$ algorithm finishes in constant time independent of N although this constant diverges as $\alpha \to 1$. The key fact [4] about superlinear networks is that there is a cutoff

$$k_{\max} = \alpha/(\alpha - 1) \tag{35}$$

such that only a finite number of nodes have more than $k_{\max}$ connections. By choosing t_0 sufficiently large, no nodes $t > t_0$ will have more than $k_{\max}$ connections. We will show that the running time of the parallel part of the algorithm is roughly $k_{\max}$ steps.

Consider what happens on the first step of the algorithm. All nodes $t > t_0$ are initially connected to the gel node so the leading behavior of the normalization is $Z^1(t) \sim t^\alpha$ and the leading behavior of the connection probabilities, defined in Eq. (21), is

$$p_n^1(t) \sim t^{-\alpha} \tag{36}$$

for $n \neq g$. Summing over all $n \neq g$, we find that on the first step, the probability that node t will connect away from the gel node behaves as $t^{1-\alpha}$. The expected change in the number of nodes connecting to the gel node during step one is obtained by summing over all nodes $t_0 < t < N$, with the result

$$\delta k_g^2(N) \equiv k_g^1(N) - k_g^2(N) \sim N^{2-\alpha}. \tag{37}$$

If $\alpha > 2$, no changes are expected to occur in the first step of the algorithm and we are done. This result is consistent with the fact that for $\alpha > 2$, there are only a finite number of nodes with more than one connection and these are all determined before t_0.

For $1 < \alpha < 2$, additional steps are needed before $\delta k_g^S(N)$ is less than 1 and the algorithm is done. We make the ansatz that

$$\delta k_g^S(t) \sim t^{\gamma(S)} \tag{38}$$

and look for a self-consistent solution for $\gamma(S)$. The running time T is obtained by solving for the least T such that $\gamma(T) < 0$.

On the second and later steps of the algorithm, the conditional connection probabilities, defined in Eqs. (21)–(23), can be written to leading order and for $n \neq g$ as

$$\rho_n^S(t) = \frac{[Z^{S-1}(t)/Z^S(t)]k_n^S(t)^\alpha - k_n^{S-1}(t)^\alpha}{k_g^{S-1}(t)^\alpha}. \tag{39}$$

There are two ways for $\rho_n^S(t)$ to be nonzero. The first is for there to have been a new connection from t' to n, with $n < t' < t$, in step $S - 1$. The expected number of nodes, n, that received new connections in step $S - 1$ is just $\delta k_g^S(t)$. Since $k_g^S(t) \sim t$ for all S, the leading behavior of $\rho_n^S(t)$ is $t^{-\alpha}$ and the overall probability that t will connect away from the gel node by this mechanism scales as $\delta k_g^S(t)t^{-\alpha} \sim t^{\gamma(S)-\alpha}$.

The second way for $\rho_n^S(t)$ to be nonzero is for the ratio $Z^{S-1}(t)/Z^S(t)$ to exceed unity. This possibility applies for all target nodes, $n < t$, $n \neq g$. The leading behavior of this ratio is given by

$$Z^{S-1}(t)/Z^S(t) \sim \frac{[k_g^S(t) + \delta k_g^S(t)]^\alpha}{k_g^S(t)^\alpha} \sim 1 + \alpha \frac{\delta k_g^S(t)}{k_g^S(t)} \tag{40}$$

so that the leading behavior of $\rho_n^S(t)$ is $\delta k_g^S(t)t^{-(1+\alpha)}$. Since there are t target nodes, the total probability that t will connect away from the gel node by this mechanism again scales as $t^{\gamma(S)-\alpha}$.

Combining both of the above mechanisms for t to connect away from the gel node and summing over all t, $t_0 < t < N$ (and still connected to the gel node), we obtain an expression for $\delta k_g^{S+1}(N)$, the expected number of nodes directed away from the gel node on step S,

$$\delta k_g^{S+1}(N) \sim N^{\gamma(S)+1-\alpha}. \tag{41}$$

Using the ansatz of Eq. (38), we obtain the recursion relation,

$$\gamma(S+1) = \gamma(S) + 1 - \alpha. \tag{42}$$

The recursion relation and the initial condition $\gamma(2) = 2 - \alpha$, Eq. (37), has the solution

$$\gamma(S) = \alpha - (\alpha - 1)S. \tag{43}$$

The running time of the algorithm is obtained from the least T for which $\gamma(T)$ is negative,

$$T = \alpha/(\alpha - 1), \tag{44}$$

or, from, Eq. (35),

$$T = k_{\max}. \tag{45}$$

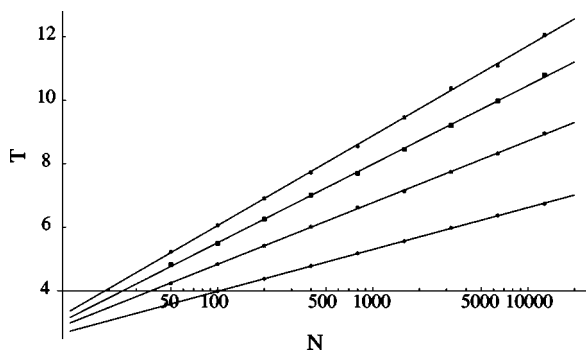This result can be understood in terms of the following sequence of events for creating connections for nodes beyond

FIG. 1. The average parallel time T to generate a network as a function of system size N for $\alpha = 0.25$, 0.5, 0.75, and 1.0, from bottom to top, respectively.
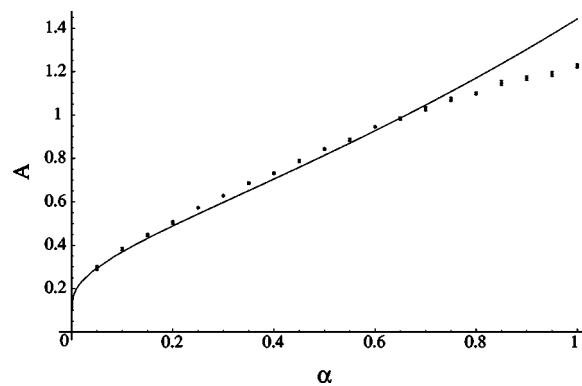

FIG. 2. The coefficient A of the leading logarithmic term in the running time vs α. The points are the results of the simulation and the solid line is the theoretical approximation, Eq. (34).

t_0. In the first parallel step, almost all nodes with two connections are generated. In the second parallel step, a small fraction of these nodes develop a third connection and a comparable number of nodes with one connection get a second connection. On the third step, an even smaller number of nodes with three connections get a fourth connection and so on until nothing happens. Note that the analysis of the algorithm reproduces the results in [4] for the scaling of the number of nodes with 2, 3, …, $k_{\max}$ connections.

VI. SIMULATION RESULTS FOR LINEAR AND SUBLINEAR KERNELS

In Sec. V A, we argued that the algorithm for the sublinear kernel requires logarithmic parallel time to generate a network, and in Eq. (34) we estimate the coefficient of the logarithm. In this section, we support these conclusions with a simulation of the parallel algorithm on a single processor workstation. In the simulation, the work of each processor on the PRAM is done in sequence making sure not to update the database describing the network until a parallel step is completed. We generated 1000 networks for each value of α and for each system size. Values of α ranged from 0 to 1, in increments of 0.05 and system sizes from 50 nodes to 12 800 nodes with each size separated by a factor of 2. Figure 1 shows the average number of parallel steps versus system size for $\alpha = 0.25$, 0.5, 0.75, and 1.0. The figure demonstrates the logarithmic dependence of average running time T on system size N for all values of α and the full range of system sizes so that that, to a good approximation, $T = A(\alpha) \log N$. Figure 2 shows a plot of the coefficient A as a function of α. The results are plotted for $0 \leq \alpha \leq 1$. The prediction of Eq. (34) is shown in the same figure. Although not perfect, the approximation of Eq. (34) captures the general trend of the data and is within a few percent of the numerical results for $\alpha < 0.8$. The larger fluctuations in connectivity near $\alpha = 1$ may explain why the "mean-field" assumption underlying the theoretical curve loses accuracy there. The theoretical estimate does appear to correctly predict that $A(\alpha)$ approaches zero with infinite slope as $\alpha \to 0$.

VII. DISCUSSION

We have examined the parallel computational complexity of generating networks obeying preferential attachment growth rules. We demonstrated that these networks can be sampled in parallel time that is much less than the size of the network. This result is surprising because the defining rules for generating these networks are sequential with nodes added to the network one at a time depending on the present state of the network. Nonetheless, we have bounded the depth of sampling growing networks by exhibiting efficient parallel algorithms for the three cases, $0 \leq \alpha < 1$, $\alpha = 1$, and $\alpha > 1$. The average parallel running time for the $0 \leq \alpha < 1$ algorithm is logarithmic, the algorithm for the BA scale-free network runs in $\mathcal{O}(\log \log N)$ time, and for $\alpha > 1$ the algorithm runs in constant time.

Growing networks thus provide an example of a discontinuous phase transition in complexity as a function of α at $\alpha = 1$. It is not surprising that a complexity transition occurs at $\alpha = 1$ since this is where the structural properties of the system also undergo a discontinuous transition from a high-temperature ($\alpha < 1$) regime where no nodes have a finite fraction of the connections to a low-temperature ($\alpha > 1$) regime where there is a single gel node with almost all connections. It is noteworthy that parallel time is the proper resource to observe this transition. The more common complexity measure of sequential time or computational work has no transition since it requires $\mathcal{O}(N)$ time to give an explicit description of the network for any α.

Our results set upper bounds on the depth of sampling growing networks, but we cannot rule out the existence of yet faster parallel algorithms. For example, if a constant time algorithm exists for $0 \leq \alpha < 1$, it would modify the conclusion that there is a discontinuous complexity transition at $\alpha = 1$. There are few rigorous lower bounds in computational complexity theory, so, in general, conclusions concerning the depth of sampling and the existence of complexity transitions in statistical physics must be considered tentative.

In this paper, we have presented a general strategy for parallelizing a broad class of sequential stochastic processes, exemplified by the coin toss with memory. We have applied the general method to create algorithms that efficiently parallelize preferential attachment network models. The general method should be more broadly applicable to growing network models with more complicated rules. To give one example, Hajra and Sen [16] extend the preferential attachment

model to include an aging factor [$F(k)$ becomes $F(k,t-n)$] so that older nodes are either favored or avoided depending on a parameter. Our algorithm can be modified to efficiently handle this class of models.

It is also instructive to examine a growing network model where our general method is not efficient. If $\alpha<0$, a case examined by Onody and deCastro [17], the general method can be applied but will not be efficient. The problem is that lower bounds on connection probabilities are typically extremely small and the algorithm will connect only a few nodes in each parallel step. We are currently investigating methods to efficiently parallelize $\alpha<0$ networks.

The fact that preferential attachment growing networks have no more than logarithmic depth indicates that they are not particularly complex objects. On the other hand, very complex biological and social systems generate networks with similar properties. If growing network models accurately describe the networks generated by these systems, one must conclude that the complexity and history dependence of the systems generating the networks are not manifest in the networks themselves. An alternative possibility is that the real networks are themselves complex but that growing network models lack some essential statistical properties of the real networks.

[1] M. Newman, Proc. Natl. Acad. Sci. U.S.A. **98**, 404 (2001).

[2] R. Albert and A.-L. Barabasi, Science **286**, 509 (1999).

[3] P. L. Krapivsky, S. Redner, and F. Leyvraz, Phys. Rev. Lett. **85**, 4629 (2000).

[4] P. Krapivsky and S. Redner, Phys. Rev. E **63**, 066123 (2001).

[5] J. Machta and R. Greenlaw, J. Stat. Phys. **77**, 755 (1994).

[6] J. Machta and X.-N. Li, Physica A **300**, 245 (2001).

[7] C. Moore and J. Machta, J. Stat. Phys. **99**, 661 (2000).

[8] J. Machta and R. Greenlaw, J. Stat. Phys. **82**, 1299 (1996).

[9] D. Tillberg and J. Machta, Phys. Rev. E **69**, 051403 (2004).

[10] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky, Nature (London) **400**, 133 (1999).

[11] S. N. Dorogovtsev, J. F. F. Mendes, and A. N. Samukhin, Phys. Rev. Lett. **85**, 4633 (2000).

[12] J. Machta, Am. J. Phys. **67**, 1074 (1999).

[13] C. H. Papadimitriou, *Computational Complexity* (Addison-Wesley, Reading, MA, 1994).

[14] A. Gibbons and W. Rytter, *Efficient Parallel Algorithms* (Cambridge University Press, Cambridge, 1988).

[15] R. Greenlaw, H. J. Hoover, and W. L. Ruzzo, *Limits to Parallel Computation: P-completeness Theory* (Oxford University Press, Oxford, 1995).

[16] K. Hajra and P. Sen, e-print cond-mat/0406332.

[17] R. Onody and P. de Castro, e-print cond-mat/0402315.

[18] Preferential attachment models were considered much earlier by H. A. Simon, Biometrika **42**, 425 (1955).