PHYSICAL REVIEW E **70**, 066707 (2004)

Efficient algorithm for the forest fire model

Gunnar Pruessner* and Henrik Jeldtoft Jensen†
Department of Mathematics, Imperial College London, 180 Queen's Gate, London SW7 2BZ, United Kingdom

(Received 7 September 2003; published 22 December 2004)

The Drossel-Schwabl forest fire model is one of the best studied models of nonconservative self-organized criticality. However, using an alternative algorithm, which allows us to study the model on large statistical and spatial scales, it has been shown to lack simple scaling. We thereby show that the considered model is not critical. This paper presents the algorithm and its parallel implementation in detail, together with large-scale numerical results for several observables. The algorithm can easily be adapted to related problems such as percolation.

DOI: 10.1103/PhysRevE.70.066707 PACS number(s): 02.70.−c, 64.60.Ht, 05.65.+b, 02.50.−r

I. INTRODUCTION

The assumption that self-organized criticality (SOC) [1] is the correct framework to describe and explain the ubiquity of power laws in nature has been greatly supported by the development of nonconservative models, because natural processes are typically dissipative. Contrary to these models, analytical work has suggested that the deterministic part of the dynamics must be conservative in order to obtain scale invariance [2,3]. However, on a mean-field level, this is not necessarily true [4], which has been exemplified in an exact solution of a model that has a forest-fire-like driving [5]. However, as a random neighbor model, the latter lacks spatial extension.

The Drossel-Schwabl forest fire model (DS-FFM) [6] is one of the few spatially extended, dissipative models which supposedly exhibit SOC. Contrary to the Olami-Feder-Christensen stick-slip model [7], where criticality is still disputed (for recent results, see, for example, [8–10]), for the DS-FFM the asymptotic divergence of several moments of its statistics, and therefore the divergence of an upper cutoff, can be shown rigorously. Although this might be considered as a sign of criticality, it is far from being a sufficient proof. In equilibrium thermodynamics, ‘‘criticality’’ usually refers to a divergent correlation length [11,12] in the two-point correlation function, which is associated with a scale-invariant or power-law-like behavior. This is how the term ‘‘criticality’’ is to be interpreted in SOC: Observables need to be scale invariant [52], i.e., power laws in the statistics. There are many examples of divergent moments without scale invariance, such as the overcritical branching process [13] or overcritical percolation [14].

Thus, there is *a priori* no reason to assume that the DS-FFM is scale-free. However, there are many numerical studies that suggest this [6,15,16]; one of them, however, suggests the breakdown of simple scaling [17]. Since an analytical approach is still lacking, numerical methods are required to investigate this problem. In this paper, we propose an alternative, very fast algorithm to simulate the DS-FFM with large statistics and on large scales. The implementation of the algorithm has produced data of very high statistical quality. Some of the results have been already published elsewhere [18].

The structure of the paper is as follows. Section II contains the definition of the model together with its standard observables and their relations. Then the algorithm is explained in detail. The section finishes with a detailed discussion on the changes necessary to run the algorithm on parallel or distributed machines. In Sec. III, results for the two-dimensional FFM are presented and analyzed. The paper concludes with a summary in Sec. IV.

II. METHOD AND MODEL

This section is mainly technical: After defining the model, all relevant details of the implementation are discussed. Apart from concepts such as the change from a tree-oriented algorithm to a cluster-oriented algorithm, concrete technical details are given, for example memory requirements and methods for handling histograms. The section also contains a description of the performance analysis of the implementation. A parallelized version of the algorithm is introduced and discussed in the final subsection.

A. The model

A forest fire model was first proposed by Bak, Chen, and Tang [19] and changed later by Drossel and Schwabl [6] to what is now known as *the* forest fire model (or DS-FFM as we call it): On a d-dimensional lattice of linear length L, each site has a variable associated with it, which indicates the state of the site. This can either be ‘‘occupied’’ (by a tree), ‘‘burning’’ (occupied by a fire), or ‘‘empty’’ (ash). In each time step, all sites are updated in parallel according to the following rules: If a site is occupied and at least one of its neighbors is burning, it becomes burning in the next time step. If a site is occupied and none of its neighbors is burning, it becomes burning with probability f. If a site is empty, it becomes occupied with probability p. If a site is burning, it becomes empty in the next time step with probability 1. As these probabilities become very small, they are better described as rates in a Poisson-like process. From a simple analysis, it is immediately clear [15] that the model can become critical only in the limit $p \to 0$ and $f \to 0$. In this limit,

*Email address: gunnar.pruessner@physics.org
†Email address: h.jensen@ic.ac.uk

```
FOREVER {
   /* Choose a site randomly */
   rn = random site;
   /* If empty occupy with probability p */
   IF (rn empty) THEN {
      with probability p: rn=occupied;
   } ELSE {
   /* If occupied start a fire with probability f */
      with probability f:
         burn entire cluster connected to rn;
   }
}
```

FIG. 1. The naive, basic algorithm of the DS-FFM.

the burning process becomes instantaneous compared to all other processes (see also Sec. II B 2) and can be represented by the algorithm shown in Fig. 1.

Compared to the instantaneous burning, both of the remaining processes are slow. In Sec. II B 2 it is shown that $p \gg f$ is required [15] for criticality, so that $f/p < 1$ and the algorithm in Fig. 1 can be written as Fig. 2, which is faster than the former, because the number of random choices of a site is reduced, but equivalent otherwise.

The line "with probability p" makes sure that the occupation attempt still happens with probability p and the burning attempt still occurs with $pf/p = f$. Of course, the line is completely meaningless, because the alternative, which occurs with probability $1-p$, is no action at all. It therefore can be omitted. Then every randomly picked empty site will become occupied, while burning happens with the reduced probability f/p.

This rescaling of probabilities is only possible in this form if the two processes are independent, which is the case because a new occupation can only occur for empty sites, while a burning attempt operates only on occupied sites. If both processes were to operate on the same type of site, a reduced probability $(1+f/p)^{-1}$ would decide between the two alternatives.

The implementation shown in Fig. 2 (without the meaningless line) has been used, for example, in [20,21]. However, probably for historical reasons, the model is usually [15,17,22] implemented as shown in Fig. 3, where trees are

```
FOREVER {
   /* The following line is without effect */
   with probability p: {
      rn = randomly chosen site;
      IF (rn empty) THEN {
         rn=occupied;
      } ELSE {
         with probability f/p:
            burn entire cluster connected to rn;
      }
   }
}
```

FIG. 2. A faster algorithm, doing essentially the same as the one shown in Fig. 1.

```
FOREVER {
   /* This is just a loop to occupy the
    * right number of sites */
   REPEAT p/f TIMES {
      rn = randomly chosen site;
      IF (rn empty) THEN {rn=occupied;}
   }
   rn = randomly chosen site;
   IF (rn occupied) THEN {
      burn entire cluster connected to rn;
   }
}
```

FIG. 3. The traditional implementation.

grown in chunks of p/f between two lightning attempts. Although this means that sites become reoccupied only in chunks of p/f, it turns out that apart from peaks in the histogram of the time series of global densities of occupied sites [22], the statistics do not depend on these details. In order to avoid any confusion, all data for this paper have been produced by means of the algorithm in Fig. 3. Moreover, this algorithm is much more suitable for parallelization (see Sec. II E).

B. Statistical quantities

The objects of interest in the DS-FFM are clusters formed by occupied sites: Two trees belong to the same cluster if there exists a path between them along nearest-neighboring, occupied sites. The cluster in the DS-FFM corresponds to avalanches in sandpilelike models [1]. The cluster, which is burnt at each burning step, can be examined more closely, so that various geometrical properties can be determined either as averages (and higher moments) or as entire distribution: Mass (in the following, this term is used synonymously with size), diameter, time to burn it, etc. The last property is better expressed as the maximum length for all paths parallel to the axes and fully within the given cluster, connecting the initially burnt tree and each tree within the same cluster. It is the maximum number of nearest-neighbor moves one has to make to reach all sites in the same cluster, in this sense a "Manhattan distance" [23]. As trees catch fire due to nearest neighbors only, this maximum distance is the total burning time of the entire cluster. In the definition above, the "time to burn" T_M becomes a purely geometrical property of the cluster and therefore independent of the actual implementation (see Sec. II C 4) of the burning procedure.

1. Cluster size distribution

The most prominent property of the model, however, is the size distribution of the clusters, $\overline{n}(s)$, which is the single-site normalized number density of clusters of mass s, i.e., the number of clusters of size s per unit volume. The average cluster size, i.e., the average size of a cluster to which a randomly chosen occupied site belongs, is correspondingly defined as

$$\tilde{s} = \frac{\sum_s s^2 \overline{n}(s)}{\sum_s s\overline{n}(s)}. \tag{1}$$

As indicated by the bar, $\overline{n}(s)$ denotes the *expected* distribution, i.e., something to be *estimated* from the observables. On average, the probability that a randomly chosen site belongs to a cluster of size s is then $s\overline{n}(s)$. If $n_t(s)$ denotes the cluster size distribution of the configuration at time t (see below), then one expects

$$\langle n_t(s)\rangle = \overline{n}(s), \tag{2}$$

where $\langle\,\rangle$ denotes the ensemble average [as opposed to a tilde, which denotes the average over $s\overline{n}(s;\theta)$]. Assuming ergodicity, one has

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T} A_t \to \langle A\rangle \tag{3}$$

for an arbitrary quantity A_t measured at each step t of the simulation. The limit exists for all bound observables A_t.

Regarding the time t, it is worth noting that a step in the simulation is considered completed, i.e., $t\to t+1$, if the randomly chosen site for the lightning attempt was occupied, i.e., the attempt was successful, so that T is the number of burnt clusters. For sufficiently large systems, the changes of the system due to growing or lightning are almost negligible, and so are the differences between averages taken over all lightning attempts or all *successful* lightning attempts. Also, the distributions found directly before and directly after burning tend to the same expectation value for sufficiently large systems, see Sec. III A 1. It is noted only for completeness that in this paper the cluster size distribution $n_t(s)$ has been measured directly *after* the burning procedure. Therefore, $n_t(s)$ does not include the cluster burnt at time step t, just like $n_{t+1}(s)$ does not in an implementation, where the distribution is measured *before* burning.

Introducing

$$\overline{\rho} = \sum_{s=1} s\overline{n}(s) \tag{4}$$

as the average density of occupied sites, the expected distribution of burnt clusters is $s\overline{n}(s)/\overline{\rho}$. To see this, $\mathcal{P}^b_t(s)$ is introduced, denoting the distribution of clusters burnt in the tth step of the simulation. This distribution contains only one nonzero value for each t, namely $\mathcal{P}^b_t(s)=1$ for the size s of the cluster burnt at time t, and $\mathcal{P}^b_t(s)=0$ for all other s. Therefore,

$$\sum_{s=1}^{N} \mathcal{P}^b_t(s) = 1, \tag{5}$$

where N is the number of sites in the system, and $N=L^d$, which is also the maximum mass of a cluster. Since the site where the fire starts is picked randomly, the cluster burnt in time step $t+1$ is drawn randomly from the distribution $n_t(s)$ with a probability proportional to the mass of the cluster. The normalization of the distribution $s\overline{n}(s)$ is given by Eq. (4), so that for t large enough, the effect of the initial condition can be neglected,

$$\langle \mathcal{P}^b_t(s)\rangle = s\overline{n}(s)/\overline{\rho}. \tag{6}$$

In the stationary state the average density of trees, $\overline{\rho}$, is related to $\tilde{s}$ by [15]

$$\tilde{s} = \frac{1-\overline{\rho}}{(f/p)\overline{\rho}}. \tag{7}$$

This equation, as well as Eq. (6), is strictly only exact if the density of occupied sites is constant over the course of the growing phase. For very large system sizes, Eq. (7) holds almost perfectly, as shown in Table III; however, note the remarks in Sec. III A 1.

For a coherent picture $\mathcal{P}^a_t(s)$ is introduced, which is the histogram of *all* clusters, i.e., $\Sigma_s \mathcal{P}^a_t(s)$ is the number of clusters in the system at time t. According to the definition of $\overline{n}(s)$, it is

$$\langle \mathcal{P}^a_t(s)\rangle = N\overline{n}(s), \tag{8}$$

and correspondingly

$$\rho_t = \frac{1}{N}\sum_s s\mathcal{P}^a_t(s) \tag{9}$$

with $\langle\rho_t\rangle=\overline{\rho}$. Since Eqs. (6) and (8) differ on the right-hand side (RHS) only by constants rather than by random variables, both distributions $\mathcal{P}^b_t(s)$ and $\mathcal{P}^a_t(s)$ are estimators of the expected distribution $\overline{n}(s)$. Clearly, the burnt cluster distribution $\mathcal{P}^b_t(s)$ is much sparser than $\mathcal{P}^a_t(s)$, and the estimator for $\overline{n}(s)$ derived from this quantity is therefore expected to have a significantly larger standard deviation. On the other hand, its autocorrelation time is expected to be considerably smaller than that of $\mathcal{P}^a_t(s)$, because on average only $p/f+1$ entries ($\overline{\rho}p/f$ sites are occupied in each "growing loop," which is repeated on average $1/\overline{\rho}$ times) of the latter are changed between two subsequent measurements, corresponding to the number of newly occupied sites plus the cluster which is burnt down. So, $\mathcal{P}^a_t(s)$ provides a much larger sample size, but is also expected to be much more correlated. In order to judge whether it is wise to spend CPU time on calculating the full $\mathcal{P}^a_t(s)$ rather than only $\mathcal{P}^b_t(s)$, as was done in the past [15], these competing effects need to be considered by calculating the estimate for the standard deviation of the estimator of $\overline{n}(s)$ from both observables, which is discussed in detail in Sec. II D.

2. Time scales

In order to obtain critical behavior in the FFM, a double separation of time scales is required [24],

$$f \ll p \ll \left(\frac{f}{p}\right)^{\nu'}, \tag{10}$$

with some positive exponent ν'. The left relation, $f \ll p$, entails $f/p \to 0$ and therefore (10) entails $p \to 0$ and $f \to 0$. This is also the case for

$$f \ll p \ll 1, \tag{11}$$

and therefore leads to the same prescription to drive the system, however (10) entails (11) but not vice versa. This can be seen by noting that (10) entails the nontrivial relation $p^{1+1/\nu'} \ll f \ll p$. Some authors, however, just state (11) [4,17]. The three scales involved are due to three different processes and their corresponding rates.

(i) The time scale on which the burning happens, the typical time of which is handwavingly estimated as the average number of sites in a burnt cluster, $\tilde{s} \propto p/f$. A more appropriate assumption is that the typical burning time scales like a power ν of the average cluster size [24]. This should be distinguished from the scaling of the *average* time it takes to burn a cluster, because the *typical* time represents the characteristic scale of the burning time distribution, which might be very different from its average.

(ii) The time scale of the growing, which is $1/p$.

(iii) The time scale of the lightning, $1/f$.

Burning must be fast compared to growing, so that clusters are burnt down before new trees grow on the edges [24], i.e., $(p/f)^{\nu'} \ll 1/p$ or $(f/p)^{\nu'} \gg p$. In order to obtain divergent cluster sizes, growing must be much faster than lightning, i.e., $p \gg f$. Thus, the double separation reads as stated in (10). By making the burning instantaneous compared to all other processes, the dynamics effectively loses one time scale. In this case, the rates f and p, measured on this microscopic time scale, vanish, i.e., $f=0$ and $p=0$, so that the right relation of (10) is perfectly met, provided that p/f does not vanish. However, the ratio f/p remains finite and $f \ll p$ is still to be fulfilled. A finite f/p means that one rate provides a scale for the other. Measuring the rates on the macroscopic time scale, defined by the sequence of burning attempts, f becomes 1 in these new unities and p becomes $p/f \equiv \theta^{-1}$. The notation $\theta = f/p$ corresponds to [4], which is, unfortunately, the inverse of θ used in [17]. Equation (10) then means $\theta \to 0$. At first sight, this result seems paradoxical, since $\theta = 0$ is incompatible with instantaneous burning's compliance with $p \ll \theta^{\nu'}$. However, this problem does not appear in the *limit* $\theta \to 0$. In a finite system, one cannot make θ arbitrarily small, as the system will asymptotically oscillate between the two states of being completely filled and completely empty. On the other hand, for fixed θ and sufficiently large system sizes, a further increase in system size will leave the main observables, such as ρ_t and $\mathcal{P}^a$ (see Sec. II B 1), essentially unchanged. These asymptotic values, namely the observables at a given θ in the thermodynamic limit, are to be measured.

3. Scaling of the cluster size distribution

Assuming that finite-size effects do not play any role, i.e., for θ not too small, the ansatz

$$\overline{n}(s;\theta) = s^{-\tau}\mathcal{G}(s/s_0(\theta)) \tag{12}$$

as obtained in percolation [14] is reasonable for s larger than a fixed lower cutoff. In the following, the additional parameter θ in $\overline{n}(s;\theta)$ is omitted, whenever possible. The quantity $s_0(\theta)$ is the upper cutoff and is supposed to incorporate all θ dependence of the distribution. It can be shown easily [15] that the second moment of $\overline{n}(s;\theta)$ [see Eq. (1)] diverges in the limit $\theta \to 0$ and $L \to \infty$, so that s_0 must diverge with $\theta \to 0$. Here, $\mathcal{G}(x)$ plays the role of a cutoff function, so that $\lim_{x\to\infty}\mathcal{G}(x)=0$ and falls off faster than any power of x for large x, because all moments of $\overline{n}(s;\theta)$ are finite in a finite system. For finite x, $\mathcal{G}(x)$ can show any structure and does not have to be constant. However, assuming $\lim_{s_0\to\infty}\overline{n}(s;\theta)$ finite, $\mathcal{G}(s/s_0)$ can be regarded as constant in s for sufficiently large s_0, so that $\overline{n}(s;\theta)$ behaves like a power law, $s^{-\tau}$, for certain s. However, *a priori* it is completely unknown whether s_0 is large enough in that sense, and the *only* way to determine τ directly from $\overline{n}(s;\theta)$ is via data collapse. It is already known that "simple scaling" (12) does not apply in the presence of finite-size effects [22].

The assumption (12) states that the FFM is scale-free in the limit $s_0(\theta) \to \infty$ and *defines* the exponent τ which characterizes the scale invariance. One cannot stress enough that with the breakdown of Eq. (12), the proposed exponent is undefined, unless a new scaling behavior is proposed. It has been pointed out that Eq. (12) certainly contains corrections [25]. This asymptotic character of the universal scaling function is well known [26] from equilibrium critical phenomena.

While Grassberger concludes that the ansatz (12) "cannot be correct" [17], this is rejected in [22]. However, the latter authors do not actually investigate $\mathcal{G}(x)$ and simply plot their estimate of $s\overline{n}(s;\theta)$ versus $s/s_0(\theta)$. In the results section, it is shown that there is no reason to believe that Eq. (12) could hold in any finite system.

4. Other distributions

The exponent τ as defined in Eq. (12) can be related to exponents of other assumed power laws. To this end, the distribution $\mathsf{P}(s,T_M;\theta)$ is introduced, which is the joint probability density function (PDF) for a cluster burnt to be of mass s and burning time (see Sec. II B) T_M at given θ. Then it is possible to define conditional expectation values as [27]

$$\mathsf{E}(s|T_M;\theta) = \sum_{s'} s'\,\mathsf{P}(s',T_M;\theta), \tag{13}$$

$$\mathsf{E}(T_M|s;\theta) = \sum_{T_M'} T_M'\mathsf{P}(s,T_M';\theta). \tag{14}$$

Moreover, it is clear that $\overline{n}(s;\theta)$ is just a marginal distribution, i.e.,

$$s\overline{n}(s;\theta) = \sum_{T_M'} \mathsf{P}(s,T_M';\theta) \equiv \mathsf{P}_s(s;\theta). \tag{15}$$

In the assumed absence of any scale, it is reasonable to define for the distribution of T_M similar to Eq. (12),

$$\mathsf{P}_{T_M}(T_M;\theta) = T_M^{-b}\mathcal{G}_{T_M}(T_M/T_{M_0}(\theta)) \tag{16}$$

and for the relation between $\mathsf{E}(s|T_M)$ and T_M,

$$\mathsf{E}(s|T_M) \propto T_M^{\mu'}. \tag{17}$$

To avoid confusion, it is important to keep in mind that the absence of scales is not a physical or mathematical necessity: The system could as well "self-organize" to any other, sufficiently broad, distribution that could have an intrinsic, finite scale, i.e., a natural constant characterizing the features of the distribution. This looks much less surprising considering the fact that standard models of critical phenomena [12], such as the Ising model, possess such a scale everywhere apart from the critical point.

An additional assumption is necessary in order to produce a scaling relation,

$$\mathsf{P}_{T_M}(T_M;\theta)dT_M = \mathsf{P}_s(\mathsf{E}(s|T_M);\theta)d(\mathsf{E}(s|T_M;\theta)), \tag{18}$$

where P_{T_M} and P_s denote the marginal distributions of $\mathsf{P}(s,T_M;\theta)$, which leads—assuming sufficiently large s_0 and T_{M_0}—to

$$b = 1 + \mu'(\tau - 2) \tag{19}$$

using $\mathsf{P}_s = s\overline{n}(s;\theta)$ and Eq. (12). Equation (18) is based on the idea that a cluster requiring burning time T_M is as likely to occur as a cluster of the size corresponding to the average taken conditional to the burning time T_M. If the distribution $\mathsf{P}(s,T_M;\theta)$ is very narrow, such that $\mathsf{E}(s|T_M)$ is virtually the only value of s with nonvanishing probability [53], this condition is met. However, the distribution can have any shape and still obey the assumption, as illustrated in Fig. 4.

Scaling relation (19) can only be derived via Eq. (18), which cannot be mathematically correct, as P_s is actually only defined for integer arguments, while in general $\mathsf{E}(s|T_M)$ is not integer valued. However, the scaling relation might hold in some limit.

The exponent defining the divergence of s_0 in Eq. (12) is defined as

$$s_0(\theta) = \theta^{-\lambda}, \tag{20}$$

leading together with Eqs. (1) and (7) to the scaling relation [24]

$$\lambda(3-\tau) = 1. \tag{21}$$

The corresponding exponent for T_{M_0} in Eq. (16) is

$$T_{M_0}(\theta) = \theta^{-\nu'}. \tag{22}$$

The assumption $T_{M_0} = \mathsf{E}(T_M|s_0) \propto s_0^{1/\mu'}$ then gives the scaling relation

$$\nu' = \frac{\lambda}{\mu'}. \tag{23}$$

It is interesting to note that this assumption is consistent with the assumption that clusters that have a size of the order

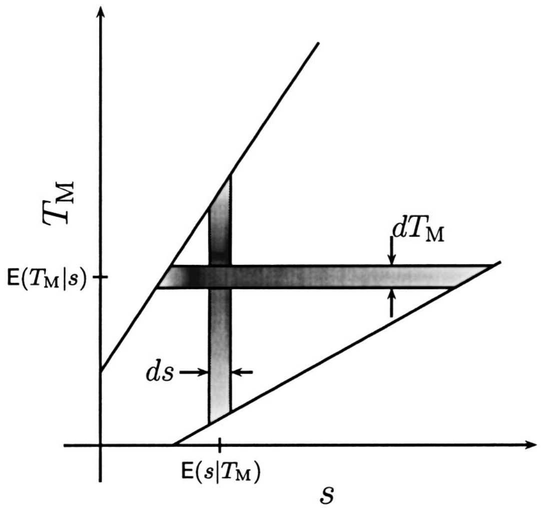

FIG. 4. A schematic joint PDF $\mathsf{P}(s,T'_M;\theta)$. The gray shading is used to indicate the density and the straight lines indicate roughly the limits of the distribution. While a narrower distribution would most easily obey Eq. (18), it does not necessarily have to be sharply peaked. In this example, the weighted areas of the horizontal and the vertical stripes might be the same. They cross at the conditional averages.

$s_0(\theta)$ need of the order T_{M_0} time to burn. In that case, one has $\mathsf{P}_{T_M}(T_{M_0};\theta)dT_M = \mathsf{P}_s(s_0;\theta)ds$, and as $T_{M_0} \propto s_0^{\nu'/\lambda}$, one has using Eqs. (16) and (12)

$$(1-b)\frac{\nu'}{\lambda} = 2 - \tau \tag{24}$$

corresponding to Eq. (19) with Eq. (23).

C. The implementation

In this section, a new implementation of the DS-FFM is discussed. An implementation especially capable of handling large scales has been proposed by Honecker [28] earlier. Its most prominent feature is the bitwise encoding of the model, which significantly reduces memory requirements. Some of the properties investigated profit from this scheme of bitwise encoding, because bitwise logical operators can be used to determine, for example, correlations and operate on entire words "in parallel." However, in this implementation it would have been inefficient to count all clusters, i.e., $\overline{n}(s)$ is determined via $\mathcal{P}^b(s)$ rather than $\mathcal{P}^a(s)$.

In contrast to standard implementations [15,20,22], where $\overline{n}(s)$ is derived from $\mathcal{P}^b(s)$, the philosophy of the implementation presented in this paper is to count *all* clusters efficiently by keeping track of their growing and disappearance, so that $\overline{n}(s)$ is derived from $\mathcal{P}^a(s)$. By comparing the standard deviation of the estimates, and the costs (CPU time), the efficiency is found to be at least one order of magnitude higher. At the same time, the complexity of the algorithm is essentially unchanged, namely $O(\theta^{-1}\log(N))$ instead of $O(\theta^{-1})$, while a naive implementation of the counting of all clusters is typically of order $O(N)$. In the following, the algorithm is described in detail. Because of its

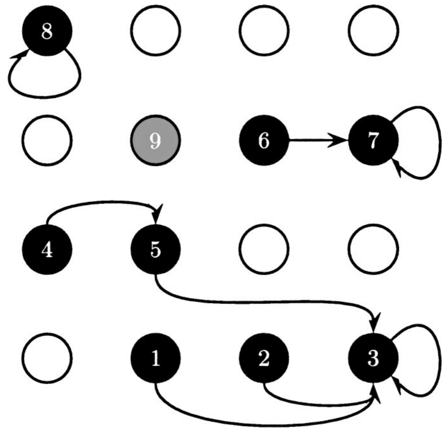

FIG. 5. All occupied sites (black) on the lattice point to a representative. The site pointing to itself is the root of the cluster. The site shown in light gray is the one which is about to become occupied, as shown in Fig. 7. The labels on the sites are just to uniquely identify them in other figures.

close relation to standard percolation, the algorithm presented below is also applicable for this classical problem of statistical mechanics. In fact, the percolation algorithm recently proposed by Newman and Ziff [29,30] is very similar. Based on many principles presented in this paper, an asynchronously parallelized version for percolation has been developed recently [31].

1. Tracking clusters

Usually each site is represented by a two-state variable, which indicates whether the site is occupied or empty. The variable does not need to indicate the state "burning," because the burning procedure is instantaneous compared to all other processes and can be implemented without introducing a third state (see Sec. II C 4). In order to keep track of the cluster distribution, each site gets associated two further variables (in an actual implementation, the number of variables can be reduced to one, see Sec. II C 2), one which points (depending on the programming language either directly as an address or as an index) to its "representative" and one which contains the mass of the cluster to which the given site is connected. The representative of a site is another site of the same cluster, but not necessarily and in fact typically not a nearest neighbor. This is shown in Fig. 5. If a site is empty, the pointer to a representative is meaningless. The pointer of representatives forms a treelike structure, because representatives might point to another representative, as shown in Fig. 6. A site which points to itself and is therefore its own representative is called a "root" site, since it forms the root of the treelike structure. Only at a root site is the second variable, denoting the mass of the cluster, actually meaningful and indicates the mass of the entire cluster. Each cluster is therefore uniquely identified by its root site: Any two sites that belong to the same cluster have the same root and vice versa. By construction of the clusters (shown below), it takes less than $O(\log N)$ to find the root of any site in the system.

The algorithm is a dynamically updated form of the Hoshen-Kopelman algorithm [32]. The same technique has

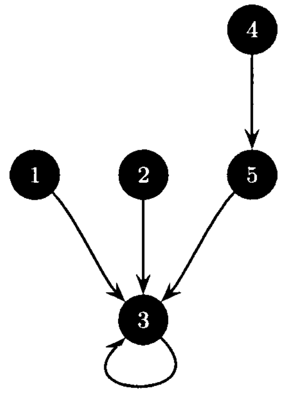

FIG. 6. The treelike structure of the largest cluster in Fig. 5.

recently been used to simulate percolation efficiently for many different occupations densities [29]. The method described in the following differs from [29] by not only growing clusters but also by removing them. While one of the main advantages of the original Hoshen-Kopelman algorithm is its strong reduction of memory requirements to $O(L^{d-1})$, the algorithm described here only makes use of the data representation proposed by Hoshen and Kopelman, so that the memory requirements are still $O(L^d)$.

In the following, the technique of how to create and to update the clusters is described in detail.

Starting from an empty lattice, the first site becomes occupied by setting the state variable. Since this site cannot be a member of a larger cluster, its representative is the site itself. Therefore, the mass variable must be set to 1. The same pattern applies to all other sites that get occupied, as long as they are isolated. The procedure becomes more involved when a site induces a merging of clusters. This is the case whenever one or more neighbors of the newly occupied site are already occupied. In general, the procedure is then as follows: (i) Find the root of all neighboring clusters; (ii) reject all roots that appear more than once in order to avoid double counting; (iii) identify the largest neighboring cluster; (iv) increase the mass variable of the root of this cluster by the mass of all remaining clusters (ignoring those which have been rejected above) plus one (for the newly occupied site); (v) bend the representative pointers of the roots of all remaining clusters to point to the root of the largest cluster (this keeps the tree height small, see below); and (vi) bend the representative pointers of the newly occupied site to point to the root of the largest cluster.

This procedure is depicted in Fig. 7, illustrating the joining of the clusters shown in Fig. 5. As an optimization, one could also bend the pointer of site 6 to point to site 3, which would effectively be a form of path compression. However, as shown below, the trees generated have only logarithmic height, so that the path compression possibly costs more CPU time than it saves for system sizes reachable with current computers [54]. It is important to note that only the root of the largest cluster is not redirected.

To find the root of a given site, which is necessary whenever clusters are considered for merging, an algorithm like the one shown in Fig. 8 needs $O(h_m(M(\mathcal{C})))$ time (worst case), where $h_m(M(\mathcal{C}))$ is the maximum height of a cluster

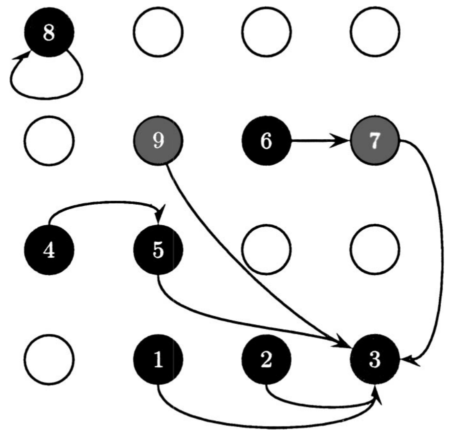

FIG. 7. The configuration in Fig. 5 after occupying the highlighted site. Sites, the pointer of which have been changed, are shown in dark gray (sites 7 and 9).

containing $M(\mathcal{C})$ sites, $\mathcal{C}$ being the cluster under consideration.

All clusters are constructed by merging clusters, which might often involve single sites. These clusters are represented as trees, like the one shown in Fig. 6. In the following, this representation is used. By construction, if at least two trees join, the resulting tree has either the height of the tree representing the largest cluster or the height of any of the smaller trees plus one—whichever is larger. Thus, by construction,

```
/* Find the root of the cluster identified
   by start_index.
 * All sites are expected to have a pointer to their
 * representative in the array pointer_of. The result
 * is stored in index. */
index = start_index
WHILE ( index != pointer_of[index] ) {
  index=pointer_of[index]; }
```

FIG. 8. The find_root algorithm. All sites are expected to have a pointer to their representative in the array pointer_of. The result of this procedure is index.

$$h_m(M) \geqslant h_m(M') \quad \text{for any} \quad M \geqslant M', \tag{25}$$

so in order to find the maximum height of a tree of mass M, one has to consider the worst case when the smaller trees have maximum height. For a given fixed M, this is the case when only two clusters merge, so

$$h_m(M) \leqslant \max(\max[h_m(M-M')|0 \leqslant M' \leqslant \lfloor M/2 \rfloor],\ \max[1+h_m(M')|0 \leqslant M' \leqslant \lfloor M/2 \rfloor]), \tag{26}$$

where $\lfloor M/2 \rfloor$ denotes the integer part of $M/2 \geqslant 0$, which is the maximum size of the smaller cluster. The outer max picks the maximum of the two max running over all allowed sizes of the smaller cluster. Using (25),

$$h_m(M) \leqslant \max(h_m(M-1), 1+h_m(\lfloor M/2 \rfloor)) \tag{27}$$

so that

$$h_m(M) \leqslant \begin{cases} 1+h_m(\lfloor M/2 \rfloor) & \text{for } 1+h_m(\lfloor M/2 \rfloor) \geqslant h_m(M-1) \\ h_m(M-1) & \text{otherwise.} \end{cases} \tag{28}$$

With $h_m(1)=1$, one can see immediately that

$$h_m(M) \leqslant \lceil \log_2(M) \rceil \tag{29}$$

by induction, noting that $\lceil \log_2(M/2) \rceil = \lceil \log_2(M) \rceil - 1$, where $\lceil a \rceil \equiv \lfloor a \rfloor + 1$ for any $a \geqslant 0$. Hence,

$$h_m(M) \in O(\log(M)), \tag{30}$$

which is therefore the (worst case) complexity of the algorithm. It is worthwhile noting that all the algorithms considered are just one solution of the more general union-find (and also insert) problem [33].

As the tree constructed is directed, there is no simple way to find all sites which are pointing to a given site. This means that splitting trees is extremely expensive in terms of complexity. However, in the DS-FFM, trees do not get removed individually, but always as complete clusters. Thus, no part of the tree structure needs to be updated during the burning (see Sec. II C 4).

2. Reducing memory requirements

The three variables (state, pointer, size) mentioned above would require a huge amount of memory: At least a bit for the state (but for convenience a byte), a word for the address, and a word for the mass (actually depending on the maxi-

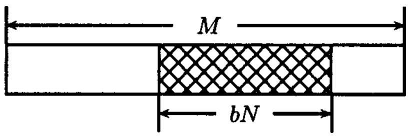

FIG. 9. The memory layout when using addresses as pointers to representative. The hatched area is used for valid addresses; what remains can be used to represent cluster masses, i.e., if the value of an address points into the white area, the value is interpreted as a mass.

mum size of the clusters). However, as the pointers are only meaningful if the site is occupied, the representative pointer can also be used to indicate the state of a site: If it is 0 (or NULL if it is an address), the site is empty and occupied otherwise.

As mentioned above (Sec. II C 1), the mass variable is meaningful only at a root site. Since only a certain range of pointers is meaningful, the remaining range can be used to indicate the mass of a cluster. Assuming that indeces can only be positive, negative numbers for the value of the pointer can be interpreted as self-references and their modulus as total mass of the cluster. The concept is restricted to system sizes that are small enough that the space not occupied by meaningful pointers is large enough to store the mass information. How large is the maximum representable system size (not to be confused with memory requirements, which is N times word size)? For a word size of $b=4$ byte, i.e., $M=2^{8b}$ representable values in a word, the maximum system size is $N=2^{31}-1$, namely $-1\cdots-N$ values for indicating masses, $1\cdots N$ for indices, and 0 for the empty site, summing up to $2N+1\leqslant M$, which is overruled by the memory required, $bN\leqslant M$, as M is (usually) the maximal addressable memory for a single process.

When using addresses as pointers, it is less obvious how to identify the range of meaningless pointers which could be used to store the mass information. In order to distinguish quickly whether a given value is an address or a mass, the most obvious way is to use higher bits in the pointers. What is the range of meaningless addresses? The addresses are words, occupying bN bytes. If each byte is individually addressable (as usual), their value differs by b, i.e., they span a range of bN different values. As shown in Fig. 9, the largest remaining continuous chunk of values, not used for references to representatives, has therefore at least size $\lceil(M-bN)/2\rceil=(M-bN)/2$, assuming that the pointer values used, which is also the range of addresses where they are stored, spans a continuous range. If the $N+1$ different cluster masses are to be represented as pointer values pointing into the meaningless region, one has $1+N\leqslant(M-bN)/2$, i.e., $(b+2)N+2\leqslant M$. If they do not have to be continuous, the condition is relaxed: $1+N\leqslant M-bN$. Alternatively one can make use of the lower bits: If the pointers point to words in a continuous chunk of memory or at least are all aligned in the same way, then all pointers are identical (mod b), i.e., all pointers p obey $p=c$ (mod b), where $0\leqslant c<b$ is a constant. Since $b>1$, one can use $p\neq c$ (mod b) to indicate that a given pointer value is to be interpreted as mass, which can easily be calculated via a bit-shift.

```
void *start_pointer, *root, *content;
/* start_pointer is the address of the site, the root of which
 * is to be found. root will always point to the site currently
 * under consideration, while content is always the address
 * root is pointing to.
 * The macro IS_SIZE verifies, whether the value given is a size. */

/* Initialise: Assume that start_pointer is the root and
 * read its content. */
for (content=*((void **)(root=start_pointer));

/* Test whether root's content is actually a size. */
    (!IS_SIZE(content));

/* Iterate: content is not a size, so the next candidate
 * is what root is currently pointing to.
 * Content is updated accordingly. */
    content=*((void **)(root=content)));
```

FIG. 10. An implementation of `find_root` in C using pointers to `void`.

In C it is reasonable to represent the sites as void * and interpret these as pointers to other sites, i.e., void **, so that the loop to search for a root just becomes the code shown in Fig. 10.

Representing each site by a word instead of a byte or even a bit [28] still leads to reasonably small memory requirements for typical system sizes (for instance, a system of size $N=4096\times4096$ would require 64 megabytes). Since the algorithm has an almost random memory access pattern, it is not reasonable to implement it out of core [34]. In order to simulate even larger sizes, the following representation has been implemented: At the beginning of the simulation, the entire lattice is splitted in cells so that whatever site in such a cell is occupied, it must belong to the same cluster as any other occupied site in the same cell, i.e., each site in the cell is the nearest neighbor of all other sites in the cell. On a hypercubic lattice these cells have size 2, as depicted in Fig. 11. Each site within such a cell must belong to the same cluster if it is occupied. Therefore, only one pointer is necessary to refer to its representative. On a triangular lattice, these cells would have size 3. Since a pointer can be non-

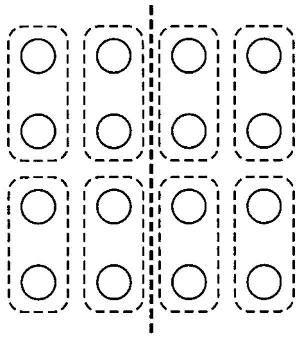

FIG. 11. If occupied, each site within a dashed box belongs to the same cluster. On a triangular lattice the dashed patches would be triangular, each one containing three sites. The thick dashed line shows the orientation of the boundary between two consecutive slices in the parallelized code, see Sec. II E.

null, although not all sites in the cell are occupied, a new variable must represent the state of the sites in each cell, if not lower or higher order bits of the pointers can be used (see above). On the hypercubic lattice, the memory requirement is therefore for each pair of sites 2 bit for the state and 1 word for the address or index of the representative. Storing the 2 bits in a byte (and keeping the remaining 6 bits unused), the memory requirements are therefore reduced to $(b+1)N/2$ bytes. Using indices, the maximum representable system size is given by $3/2N+1 \leqslant M$, and using pointers with a size identification as shown in Fig. 9, the constraint is $1+N \leqslant [M-(bN/2)]/2$ in the worst case.

3. Efficient histogram superposition

So far, only the maintenance of the cluster structure has been described. Since the masses of all clusters involved are known, it is simple to maintain a histogram of the cluster mass distribution: If a cluster of size s is burnt, the corresponding entry in $\mathcal{P}_t^a(s)$ is decreased by one. If a cluster changes size, $\mathcal{P}_t^a(s)$ is updated accordingly. For example, when two clusters of size s_1 and s_2 merge as a particular site is newly occupied during the growing procedure, $\mathcal{P}_t^a(s_1)$ and $\mathcal{P}_t^a(s_2)$ are decreased by one and $\mathcal{P}_t^a(s_1+s_2+1)$ is increased by one.

Naively, the average cluster size distribution is the average of $\mathcal{P}_t^a(s)$, i.e.,

$$\frac{1}{T}\sum_{t'=1}^{t} \mathcal{P}_{t'}^a(s), \tag{31}$$

with T the number of iterations. Depending on the resolution of the histogram, it would be very time consuming to calculate this sum for each s. Using exponential binning (which is in fact a form of hashing) in order to reduce the size of the histogram solves the problem only partly.

Ignoring any hashing, a naive superposition, where each slot in the histogram needs to be read, has complexity $O(TH)$, where H is the largest cluster mass in the histogram.

This problem is solved by noting that early changes in the histogram propagate though the entire sequence of histograms. Denoting the initial histogram as $\mathcal{P}_0^a(s)$ and $\Delta\mathcal{P}_t^a(s)=\mathcal{P}_{t-1}^a(s)-\mathcal{P}_t^a(s)$, then

$$\mathcal{P}_t^a(s)=\mathcal{P}_0^a(s)+\sum_{t'=1}^{t} \Delta\mathcal{P}_{t'}^a(s), \tag{32}$$

and therefore

$$\sum_{t=1}^{T} \mathcal{P}_t^a(s)=T\mathcal{P}_0^a(s)+\sum_{t'=1}^{T} (T-t'+1)\Delta\mathcal{P}_{t'}^a(s). \tag{33}$$

By using this identity, only the right-hand side of Eq. (33) is maintained by increasing it by $T-t+1$ when a new cluster is created at time t and by decreasing it by the same amount when it is destroyed. In this way, the complexity is only of order $O[T(\theta^{-1}+1)]$, according to the number of clusters created and destroyed, i.e., the number of changes in the distribution. This concept only becomes problematic if floating point numbers are used to store the histogram and the accuracy is so small that changes in the sum by 1 do not change the result anymore [55]. The maximum value in $\mathcal{P}_t^a(s)$, where this does not happen, is given by the largest m with $m+1 \neq m$, where m is a variable of the same type as $\mathcal{P}_t^a(s)$. For floating point number, the value of m is related to the constant DBL_EPSILON (or FLT_EPSILON for single precision), which essentially characterizes the length of the mantissa. The concrete value of m is actually platform-precision-, and type-dependent. For an unsigned integer of size 4, this value would be $(2^{32}-1)-1$, corresponding to ULONG_MAX-1; for double precision IEEE75 floating point numbers, this value is FLT_RADIX**DBL_MANT_DIG-1, i.e., $2^{53}-1$.

Provided that the right-hand side of Eq. (33) is below the threshold m discussed above for all s, this means that only a single histogram needs to be maintained. It is initialized with $T\mathcal{P}_0^a(s)$ and updated with $\pm(T-t+1)$ at time step t, when a cluster of size s appears or disappears. It is worth mentioning that this concept obviously even works in conjunction with binning (or any other hashing).

```
/* Initialise current_stack. */
CLEAR current_stack;
/* Put initial site on current_stack. */
PUT rn ON current_stack;
/* Sites are cleared right after they have entered the current_stack. */
rn = empty;
/* The first loop runs until there is nothing left to
 * burn, i.e. next_stack was not filled during the inner loop. */
DO {
   /*  Clear next_stack so that it can get filled in the next loop. */
   CLEAR next_stack;
   /* The next loop runs as long as there are sites left to burn
    * in the current generation of the fire. */
   WHILE current_stack not empty {
      /* GET: remove the upmost element from current_stack and
       * put it in x */
      GET x FROM current_stack;
      /* Visit all neighbours */
      FOR all neighbours n of x {
         if (n occupied) {
            /* Put occupied sites on the current_stack of the next
             * generation of the fire */
            PUT n ON next_stack;
            n = empty
         }
      }
   }
/*  The next current_stack to be considered is next_stack. */
   current_stack = next_stack;
} WHILE current_stack is not empty
```

FIG. 12. The burning procedure starting at rn. In an actual implementation, the copying of `next_stack` to `current_stack` can easily be omitted by repeating the code above with `current_stack` and `next_stack` interchanged, similar to a red-black approach [34].

4. Implementation of the burning procedure

The burning procedure was implemented in the obvious way, without making use of the tree structure, as shown in Fig. 12. Although the burning procedure could also be implemented explicitly recursively, it is of course significantly faster when implemented iteratively. The usage of a stack in the procedure might be thought of as reminiscent of the un-

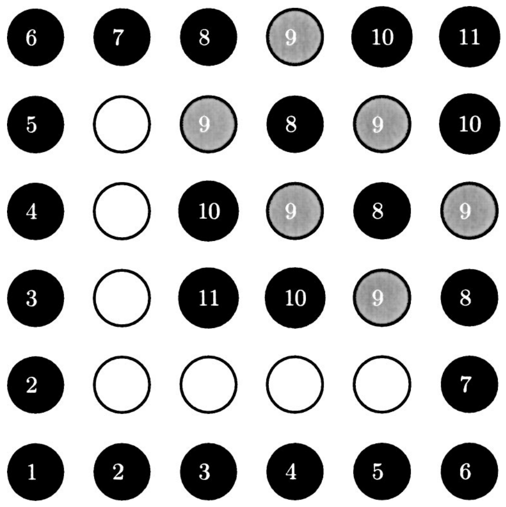

FIG. 13. The burning order for a 6×6 patch of sites, where seven sites are not occupied and form a barrier, such that some sites behind it burn later, together with the fire front propagating away from the starting point of the fire at the lower left-hand corner. The sites belonging to the largest set of trees burning at the same time are shown in light gray, unoccupied sites are shown in white, occupied sites in black. The numbers indicate the generation of the fire, which is one plus the Manhattan distance from the starting point of the fire along occupied sites.-

derlying recursive structure. The number of times the outer loop in Fig. 12 runs defines the generation of the fire front and gives T_M; other properties of the burnt cluster can be extracted accordingly. The most important resource required by this procedure is the stacks: one for the currently burning sites and one for the sites to be burnt in the next step. There is no upper limit known for the number of simultaneously burning sites, apart from the naive $N/2$ on a hypercubic lattice, which comes from the observation that sites which belong to the same generation of the fire must reside on the same sublattice (even or odd).

On the other hand, it is also trivial to find the maximal number of sites which burn at the same time, if the fire starts in a completely dense forest, i.e., in a lattice with $\rho=1$. Obviously the size of the tth generation is then given by $4(t-1)$ for $t>1$ and 1 at the beginning, $t=1$. Since the sum of these numbers is the number of sites reachable within a certain time t, *the sum* is also an upper limit for the number of simultaneously burning sites. Indeed, the actual number can easily be larger than $4(t-1)$, caused by arrangements of wholes in the lattice, which delay the fire spreading at certain sites so that they burn together with a larger fire front. Such a construction is shown in Fig. 13.

Of course it is neither reasonable nor practically possible to provide enough memory for the theoretical worst case, i.e., two stacks each of size $N/2$. Indeed the typical memory requirements seem to be of order $O(\sqrt{\theta^{-1}})$, as shown in Table I, where $f_{\max}$ denotes the largest fire front observed during the simulation. Providing stacks only of size $4L$ turned out to be a failsafe, yet pragmatic, solution. Formally one could implement a slow out-of-core algorithm in the rare yet possible case in which the memory for the stack is insufficient, i.e., use hard-disk space to maintain it. In fact, this is what *de facto* happens if one uses a stack of size $N/2$ on a virtual memory system.

TABLE I. Performance data for different parameters and setups. ‘‘ap3000,2’’ denotes a parallel run on two nodes on an AP3000, accordingly ‘‘ap3000,4’’. ‘‘cluster,10’’ denotes a cluster of 25 Intel machines, connected via an old 10 MBit network, ‘‘cluster,100’’ denotes the same cluster on a 100 MBit network. ‘‘single1’’ and ‘‘single2’’ denote two different types of single nodes. The largest fire front $f_{\max}$ was only measured on these systems. The quantity ζ is the ratio of the average time (real time on the parallel systems in order to include communication overhead, user time on single nodes) for one successful update during statistics, i.e., when all data structures need to be maintained, and equilibration (transient), i.e., when the standard representation is used.

System	L	θ^{-1}	ζ	$f_{\max}$
ap3000,2	8000	4000	1.51	
ap3000,2	8000	8000	1.52	
ap3000,4	16 000	4000	1.34	
ap3000,4	16 000	8000	1.48	
ap3000,4	16 000	16 000	1.37	
ap3000,4	16 000	32 000	1.41	
cluster,10	32 000	4000	2.71	
cluster,10	32 000	64 000	3.81	
cluster,100	32 000	32 000	1.76	
single1	1000	500	1.41	216
single1	2000	1000	1.41	326
single1	4000	125	1.42	106
single1	4000	250	1.47	172
single1	4000	500	1.48	255
single1	4000	1000	1.53	317
single1	4000	2000	1.50	518
single1	4000	4000	1.57	646
single1	4000	8000	1.48	907
single1	4000	16 000	1.45	1327
single2	8000	4000	2.11	687
single2	8000	8000	2.11	912
single2	8000	16 000	2.09	1415

5. Complexity of the algorithm

The overall complexity of the algorithm has two contributions: The ‘‘growing’’ part, where new clusters are generated from existing ones, and the ‘‘burning’’ part. The time needed for the burning part is proportional to the number of sites burnt and therefore expected as $O(\bar{s})$ [see Eqs. (1) and (7)] and $O(N)$ in the worst case. Since $\tilde{\rho}$ in Eq. (7) is bound, the complexity of ‘‘burning’’ is $O(\theta^{-1})$ (expected). The complexity of ‘‘growing’’ is estimated by the average number of sites newly occupied, θ^{-1}, times the worst-case complexity (30) to find the root of any given site, because up to four roots need to be found at each tree growing. According to Eq. (30), the worst-case complexity to find the root of any given site is $O(\log(N))$, leading to an overall complexity for ‘‘growing’’ of $O(\log(N)\theta^{-1})\supset O(\theta^{-1})$. In practice, the logarithmic correction is negligible, especially since $\log(N)$ is an extreme overestimate of the average case and therefore essentially the same runtime behavior is expected for both procedures [30]. Implementations like the one in [28] avoid this logarithmic factor by counting only the burnt cluster and therefore arrive at an overall complexity of $O(\theta^{-1})$.

The algorithm presented has therefore only a negligibly higher computational complexity compared to implementations which measure only $\mathcal{P}^b$. This is corroborated by the comparison of the CPU time per burnt cluster during equilibration, i.e., the transient, when the cluster structure does not need to be maintained and the algorithm used is the standard implementation, to the CPU time per burnt cluster during statistics, i.e., when observables are actually measured and especially $\mathcal{P}^a$ is produced. This ratio is shown as ζ in Tables I and II. It varies only slightly with L or θ^{-1}.

Apparently the algorithm presented offers more statistics, however it suffers from one limitation: It requires about $(b+1)N/2$ bytes of memory (see Sec. II C 2), compared to $N/8$ bytes in bitwise implementations like [28], i.e., typically a factor 20 more. In order to ascertain whether this disadvantage is acceptable with respect to the statistical gain, one has to determine the standard deviations of the calculated quantities for both implementations.

D. Calculating the standard deviation

In order to compare the two algorithm rigorously, it is necessary to estimate the standard deviation of the estimators for $\overline{n}(s)$ produced by them [35,36],

$$\sigma^2_{\mathcal{P}^b}(s) = \frac{2\tau_{\mathcal{P}^b}+1}{T-1}[\langle \mathcal{P}^b_t(s)^2\rangle - \langle \mathcal{P}^b_t(s)\rangle^2],$$

$$\sigma^2_{\mathcal{P}^a}(s) = \frac{2\tau_{\mathcal{P}^a}+1}{T-1}[\langle \mathcal{P}^a_t(s)^2\rangle - \langle \mathcal{P}^a_t(s)\rangle^2]. \quad (34)$$

Here $\tau_{\mathcal{P}^b}$ and $\tau_{\mathcal{P}^a}$ are the correlation times of the two quantities. Calculating the correlation time in the standard fashion by recording the history $\mathcal{P}^a_t(s)$ and $\mathcal{P}^b_t(s)$ for each s would mean storing millions of floating point numbers. Therefore, it was decided to restrict these calculations to just a small yet representative set of s values. The result shows that the standard deviation does not fluctuate strongly in s.

Because of the special form of $\mathcal{P}^b_t(s) \in 0,1$, its variance is particularly simple,

$$\langle \mathcal{P}^b_t(s)^2\rangle = \langle \mathcal{P}^b_t(s)\rangle \quad (35)$$

so that

$$\sigma^2_{\mathcal{P}^b}(s) = \frac{2\tau_{\mathcal{P}^b}+1}{T-1}\langle \mathcal{P}^b_t(s)\rangle[1-\langle \mathcal{P}^b_t(s)\rangle]. \quad (36)$$

The correlation time of $\mathcal{P}^b_t(s)$ is expected to be extremely small, not only on physical grounds—a cluster can only burn down once—but also because of the extreme dilution of $\mathcal{P}^b_t(s)$, as was described in Sec. II B 1. For fixed s, most of the $\mathcal{P}^b_t(s)$ are 0. In contrast, the $\mathcal{P}^a_t(s)$ are expected to have a large correlation time, because "only" $\theta^{-1}+1$ entries are changed between two subsequent histograms.

The correlation function is calculated in the symmetric way as proposed in [37], here for an arbitrary quantity A_t,

$$\phi^{AA}_{t'} = \frac{\langle A_t A_{t+t'}\rangle_{T-t'} - \langle A_t\rangle_{T-t'}\langle A_{t+t'}\rangle_{T-t'}}{\langle A_t^2\rangle_T - \langle A_t\rangle_T^2}, \quad (37)$$

where $\langle\,\rangle_{T-t'}$ denotes the average taken over time t from $t=1$ to $t=T-t'$. The quantity $\Phi^{AA}_{t'}$ was fitted to $\exp(-t/\tau_A)$ in order to find the correlation time τ_A. The results are given in Table II.

As described in Eqs. (6) and (8), the two estimators for $\overline{n}(s)$ differ slightly. However, except for $\overline{n}(s)$, only constant values appear on the RHS of Eqs. (6) and (8), so that the relative errors of $\langle \mathcal{P}^b_t(s)\rangle_T$ and $\langle \mathcal{P}^a_t(s)\rangle_T$ are also the relative errors of the estimators for $\overline{n}(s)$ derived from them. These relative errors are shown in Table II as well. Their ratio is given as α and is an indicator for the advantage of the algorithm proposed. If the relative error is to be improved by a factor q, one needs to invest q^2 CPU time, i.e., if the algorithm proposed in this paper costs a factor ζ more CPU time, and the gain in the relative error α, the total gain is α^2/ζ. The values for this quantity are also given in Table II.

According to the table, for fixed θ, relative errors and the correlation times are only weakly affected by an increase in system size. At first sight, this is counterintuitive, as the number of passes [20,21], i.e., the mean number of times a site has been visited between two lightnings, decreases inversely proportional to the total number of sites in the system: $1/(\theta\overline{\rho}L^2)$, see Sec. III B. Assuming that this number is essentially responsible for the error suggests keeping the number of passes constant among different L. However, this is apparently not the case, possibly because of self-averaging [38] effects.

The table also shows various tendencies, which are worth mentioning. First of all, the total gain becomes smaller for larger avalanche size s. The B in front of some of the values indicates that a bin around the s value was investigated, i.e., the time series of

$$\sum_{s'\in\mathcal{B}} \mathcal{P}^{a,b}(s') \quad (38)$$

was considered, where $\mathcal{B}$ is a set of (consecutive) s values, representing the bin. For larger values of s, these sets get exponentially larger, which is necessary for a reasonably large number of events as a basis for the estimators. The general tendency that the proposed algorithm is even more efficient at small s is not surprising: $\mathcal{P}^b$ samples from $s\overline{n}(s)$, while $\mathcal{P}^a$ samples only from $\overline{n}(s)$, i.e., $\mathcal{P}^b$ "sees" larger cluster more often. *Nevertheless $\mathcal{P}^a$ is still advantageous by roughly a factor* 5. The empty entries in Table II are due to numerical inaccuracies or simply missing simulations for certain parameters. Some entries are estimated and marked as such.

There is an additional correlation not mentioned so far: The individual points in the estimator of the distribution $\mathcal{P}^a$ are not independent. There are "horizontal correlations," i.e.,

TABLE II. Correlation times τ_b and τ_a of the corresponding observables $\mathcal{P}^b$ and $\mathcal{P}^a$ as a function of s and for different parameters L, θ^{-1}. Values of s marked by "B" are results for bins around the s value indicated. For each set of parameters, the quantity ζ is given. It denotes the ratio between the average CPU time for one successful update during equilibration (transient) and during statistics, see also Table I. The two fractions $\sqrt{\sigma^2_{\mathcal{P}^b}(s)}/\langle \mathcal{P}^b_t(s)\rangle$, $\sqrt{\sigma^2_{\mathcal{P}^a}(s)}/\langle \mathcal{P}^a_t(s)\rangle$ their ratio α and α^2/ζ are derived. A * marks cases where $\tau_b(s)=0$ has been assumed. A † marks values of $\tau_\alpha(s)$, which have been extrapolated from $\tau^\alpha(s)$ for smaller s.

L	θ^{-1}	ζ	s	$\tau_b(s)$	$\tau_a(s)$	$\frac{\sqrt{\sigma^2_{\mathcal{P}^b}(s)}}{\langle \mathcal{P}^b_t(s)\rangle}$	$\frac{\sqrt{\sigma^2_{\mathcal{P}^a}(s)}}{\langle \mathcal{P}^a_t(s)\rangle}$	α	α^2/ζ
4000	4000	1.57	10			0.0138*			
			100	0.170	23.6	0.0637	0.000 99	64.3	2633.4
			B 10^3	0.028	14.2	0.0450	0.001 91	23.6	354.8
			B 10^4	0.006	10.0	0.0412	0.004 70	8.8	49.3
			B 10^5		7.2	0.0662*	0.021 04	3.1	6.1
4000	16 000	1.45	10	0.013	39.9	0.0141	0.000 56	25.4	444.9
			100	0.126	28.8	0.0608	0.001 27	48.0	1589.0
			B 10^3	0.006	4.7	0.0457	0.001 75	26.1	469.0
			B 10^4	0.013	2.9	0.0512	0.003 32	15.4	163.6
			B 10^5		2.2	0.0433	0.007 95	5.4	20.1
8000	1000		10	0.131		0.0154			
			100	0.122	284.6	0.0602	0.001 58	38.1	
			B 10^3	0.028	236.5	0.0399	0.003 37	11.8	
			B 10^4	0.016	163.5	0.0397	0.008 78	4.5	
8000	4000	2.11	10	0.122	78.2	0.0154	0.000 52	29.8	420.9
			100	0.132	16.4	0.0634	0.000 87	72.9	2518.7
			B 10^3	0.022	8.2	0.0438	0.001 47	29.7	418.1
			B 10^4	0.005	5.5	0.0442	0.002 41	18.3	158.7
			B 10^5		4.2	0.0409*	0.010 06	4.1	8.0
			B 2×10^5		3.8†	0.0635*	0.020 55	3.1	4.6
8000	16 000	2.09	10		262.5	0.0139*	0.000 68	20.5	201.1
			100	0.131	56.1	0.0629	0.000 87	72.0	2480.4
			B 10^3	0.014	19.0	0.0467	0.001 15	40.6	788.7
			B 10^4	0.009	11.1	0.0503	0.002 96	17.0	138.3
			B 10^5	0.006	8.3	0.0411	0.006 89	6.0	17.2
			B 2×10^5		7.5	0.0423*	0.009 47	4.5	9.7
			B 5×10^5		7.0†	1.1106*	0.333 31	3.3	5.2

$\mathcal{P}^a(s)$ is correlated for different values of s. These are additional correlations due to clusters of small sizes, which are likely to grow and propagate through s in $\mathcal{P}^a_t(s)$ for consecutive time steps, i.e.,

$$\langle \mathcal{P}^a_t(s)\mathcal{P}^a_{t'}(s')\rangle - \langle \mathcal{P}^a_t(s)\rangle\langle \mathcal{P}^a_{t'}(s')\rangle. \tag{39}$$

This correlation is at least partly captured by the correlations measured for the binned data. It is to be distinguished from the correlations of *independent* realizations, where correlations are expected in the cluster size distribution also, i.e.,

$$\langle \mathcal{P}^a_t(s)\mathcal{P}^a_t(s')\rangle - \langle \mathcal{P}^a_t(s)\rangle\langle \mathcal{P}^a_t(s')\rangle. \tag{40}$$

This must be taken into account as soon as estimates of $\overline{n}(s)$ for different s are compared, as is done when an exponent is calculated by fitting. This effect is also present for $\mathcal{P}^b$, which is, however, diluted so enormously that it influences the outcome only in an insignificant way.

The horizontal correlations could be estimated using a jackknife scheme [39], similar to that used to calculate the error bar of the exponent from the time evolution of a quenched Ising model [40]. While it is certainly essential for the careful estimation of the error bar of an exponent, it is irrelevant for the discussion in this paper, as it is quantitatively based only on *local* comparisons of error bars (overlaps), while its global properties, i.e., shape and collapse with other histograms estimated, is not concerned with error bars. Some authors even seem to dismiss the relevance of these correlations completely [30].

E. Parallelizing the code

Constructing clusters and keeping track of clusters rather than of single sites seems to be in contradiction to any at-

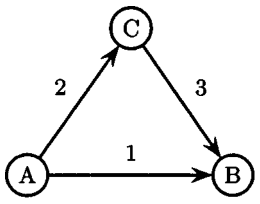

FIG. 14. Nodes *A*, *B*, and *C* send messages in the order indicated. However, it might well happen that the message sent last by node *C* to node *B*, namely message 3, arrives at that node before message 1, sent *before* message 2 was sent, which arrived *before* message 3 was sent.

tempt to run the algorithm distributed, that is, splitting the lattice into *S slices* (one-dimensional decomposition—as periodic boundaries apply, the slices may better be called cylinders). Moreover, there is a general problem of parallelization which becomes apparent in this context: The usual bottleneck of parallel systems is the communication layer. In order to keep the communication between sublattices as low as possible, fast parallel code on a lattice requires as few interactions between slices as possible, while the whole point of doing physics on large lattices is the assumption of significant interaction between their parts. It is this fundamental competition of requirement and basic assumption which makes successful parallel code so rare and which seems to indicate that problems must have very specific characteristics in order to be parallelizable in a reasonable way.

However, it is indeed possible to run the algorithm described above on parallel machines successfully in the sense that it not only makes use of the larger amount of (distributed) memory available, but also of the larger amount of computing capabilities. In fact, the code was successfully rewritten using MPI [41] and has been run on two systems with distributed memory: The massively parallel machine AP3000 at the Department of Computing at Imperial College and on a cluster of workstations (25 nodes).

In the following, the most important design characteristics are described, which proved important in order to make the code running reasonably fast. This concerns mainly the statistics part, but the equilibration also needs some tricks.

MPI assures that packets sent from one node to another in a certain order are received in exactly the same order—in the language of MPI this means that the message ordering is preserved in each individual communicator. But how different communicators relate to each other, i.e., how one stream of packets relates to another one, is not specified. If, for instance, node *A* sends a packet to node *B*, and then to node *C*, which then sends a packet to node *B*, this packet might arrive earlier at *B* than the packet first sent by *A*, see Fig. 14.

However, it is one of the main goals of parallelization to avoid any kind of synchronization, which is extremely expensive. Even in a master-slave design, as was chosen here, one encourages communication between the slaves whenever they can anticipate what to do next or can indicate to each other what to do next.

As explained above (Sec. II A), an update consists essentially of two steps: growing and burning. Both processes are now distributed among the slices. The growing procedure is realized by trying to grow θ^{-1}/S trees in each slice. This is not an exact representation of a growing procedure taking place on the entire lattice at once, because the latter has a nonvanishing probability to grow all trees at one particular spot, while the parallelized version distributes them evenly among the different slices. Provided that θ^{-1} is large compared to *S*, this effect can certainly be neglected. The advantage of the procedure is that the growing procedure at each slice does not need to be conducted by the master. The burning procedure is more complex, as the fire starts at one particular site of the entire lattice, so that it must be selected by the master. The exact procedure of the possibly following burning process depends on the stage of the algorithm.

In the following, the procedures are explained in terms of "sites" rather than "cells," as introduced in Sec. II C 2. Using cells instead of sites makes the code slightly more complicated, but the changes are obvious. If the cells are oriented parallel to the borders of slices (see Fig. 11), so that its width is a multiple of 2 in the case of a hypercubic lattice, the algorithm runs considerably faster, as the communication between the nodes is reduced by the same factor.

1. Equilibration

During the equilibration phase, it is not necessary to keep track of all clusters. Nevertheless, there is some statistics, which is very cheap to gather, namely the distribution of burnt clusters and the density of trees. The latter is very simple, as this number changes in time only by the number of grown trees minus the number of burnt trees. This is also a cross-check for the overall statistics, as the tree density is equivalent to the probability of a site to belong to *any* cluster (4).

The burning is implemented as follows: The master chooses a site from the entire lattice and sends the corresponding slice (slave) the coordinate and (implicitly) an identifier which uniquely identifies this request within this update step. The slice's response consists of the number of sites burnt (possibly 0), the identifier referring to the initial request, and possibly up to two further, new, unique identifiers. These identifiers refer to the two possible subrequests to the right and left neighboring slice due to a spreading of the fire. If a slice contacts another slice, it does so by sending the coordinates of sites which are on fire in the sending patch, together with a unique identifier. The contacted slice sends its result to the master, again together with the identifier and possibly two new ones, corresponding to the possibly two contacted neighboring slices. In this way, the master keeps track of "open (sub)requests," i.e., requests the master has been told about by receiving an answer containing information about subrequests which have not been matched by receiving a corresponding answer. The structure of requests forms a treelike structure, and if there are no open requests, the master must have received all answers of the currently burning fire. It is very important to make it impossible that by a delay of messages some answers are not counted, as it would be if the master would just count open requests, without identifying them individually. It can easily happen that the master receives an answer for a request without having received the information about the very existence of the request. It is worth mentioning that in this scheme the order of burnings is irrelevant if the burn time is not measured, as was done here.

Adding up the number of burnt sites gives the total size of the burnt cluster. This number is finally sent to all slices. If it is nonzero, the step is considered to be successful.

After equilibration, the cluster structure of pointers and roots as described above (see Sec. II C 1) needs to be constructed. This is done in a naive manner: Keeping track of sites which have already been visited, every site is visited once. The first site visited in each cluster becomes the root of all sites connected to it, which become marked as visited. The procedure corresponds to the burning procedure described above (see Sec. II C 4).

Each slice maintains a local histogram $\mathcal{P}^a$, which contain all clusters which do not have a site on the border to another slice. Otherwise, they are maintained at the master's histogram, as discussed below. In this case, the (local) root site of these clusters is moved to the border. As periodic boundary conditions apply, the only boundaries are those with other slices.

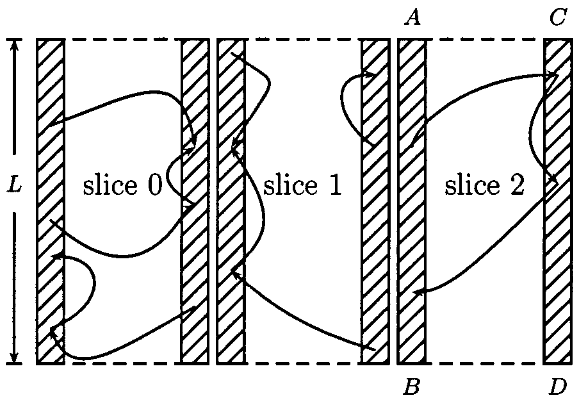

FIG. 15. The slices, three of which are shown here, maintain the references for all clusters within each slice (illustrated by arrows), even for border clusters. The references *between* slices, however, are maintained by the master. The variables $A=0$, $B=L-1$, $C=I$, and $D=I+L-1$ are the indices used for references within each slice.

2. Collecting statistics

After finishing the equilibration phase, another concept needs to be applied in order to count the total cluster size distribution $\mathcal{P}_t^a(s)$. At every update of the lattice, each slice must keep track of the clusters in the same way as was described in Sec. II C 1. Clusters which do not contain a site at a border to another slice are maintained locally, i.e., at each node has a *local histogram*. However, if a cluster contains a site at a border, it might span several slices. As soon as a cluster acquires a site at the border, it is removed from the local histogram and the site under consideration becomes the root of the cluster. The algorithm ensures that a cluster with at least one site on the border has its root at the border.

During all processes (growing or burning), the size of all clusters is updated as usual, independent of the location of the root. If the status of a border site changes, its new value or its change is put on a stack together with its coordinate. During the growing procedure, the following changes of the status are possible.

(i) *New occupation*. Change in occupation information for a site (cell). If this is the only change, then it must have been already occupied (this is only possible in an implementation using cells). If this is not the case, the reference information pointing to the root site of the given cluster must be updated also; see the next point.

(ii) *Merging border clusters*. Change of the reference information for a site (cell). This can only happen if the site (cell) was (completely) unoccupied at the time of the change or did contain size information, i.e., it was itself a root.

(iii) *General merging of clusters*. Change in size information for a site (cell). Only an increase is possible, so that any change can be represented by a single number indicating the size difference.

For each border site changing at each slice, the corresponding information is sent to the master. Typically the number of messages is not very large, because the total number of sites updated during a single growing phase is limited by θ^{-1}/S. The expected number of these messages is not given by the fraction of border sites in each slice, because changes in all border *clusters* (i.e., clusters with at least one site in the border) affect the border *sites*, as the root of each border cluster is a border site.

However, the data regarding the updates in the border do not need to be sent from the slaves to the master, if the burning attempt following the growing fails, i.e., if an empty site has been selected for lightning. Of course it is much more efficient not to send any data if not necessary. As there is only a finite number of sites in each slice, the theoretical limit of updates of border sites is bound by this number. However, it is sufficient to allocate a reasonable amount of memory ($2L$ turned out to be enough) for the stack of messages to be sent and check its limits, similar to the stack used in the burning procedure described in Sec. II C 4. Henceforth, the sending of the update information of the border is called "sending the border."

The master maintains a copy of the state of the border sites and updates a *global histogram* of border clusters. By sending the changes on the border to the master as described above, the master can update its copy of the configuration of the borders as well as the global histogram. At the end of the simulation, all histograms (S slaves histograms plus the global histogram maintained by the master node) are summed to produce the total $\mathcal{P}^a$.

As suggested in Fig. 15, the slices maintain the pointers within each slice, and these references are not changed by the master, which only connects *between* slices. If a reference at the border changes at a slice, the master receives a message to apply the corresponding changes (joining two clusters); if the size of a cluster changes, the master updates the corresponding unique root; etc. These changes are indicated by the slaves, and the master only realizes them in the copy of the border sites. Only if a change in occupation occurs must the master actually perform some nontrivial operations, because a newly occupied site might introduce a new connection *between* borders of different slices. From the point of view of the master, only borders belonging to two different, neighboring slices are directly connected and therefore to be maintained by the master, while the connectivity of the borders *within* each slice is indicated and maintained by the corresponding slave. Apart from that, the master maintains the slice spanning structures in exactly the same way as the slaves, e.g., a cluster having multiple roots among the various slices has a unique root at the master, etc.

The question arises how the master best keeps track of the changes of the borders. Ideally, a change of reference of a site at the boundary is communicated to the master simply by sending the new pointer value (index). By choosing a reasonable indexing scheme, this is indeed possible. If the value of the reference is within 0 and $L-1$, where L is the width in terms of the number of sites (or cells) (see Fig. 15), the reference denotes a site in the left border within the same slice. Similarly, if the value of a reference is within I and $I+L-1$, where I denotes the first index in the last column, a reference with such a value is bound to point to the right border of the same slice. If the master uses indexes of the range $[L,I-1]$ for denoting cross references between slices, the references are therefore unambiguous and no translation is necessary between indices used by the slices and indices used by the master.

During the burning procedure, the master can make use of its knowledge about the borders. The site selected for starting the fire is most likely a bulk size, so that the corresponding slave needs to be contacted for the occupation information. Three outcomes are possible.

(i) The site is unoccupied. Nothing happens, all slices get signaled to continue with growing.

(ii) The site is occupied, but does not contain a border site. In this case, the slice contacted can send back the size of the burnt cluster (information it knows even without actually doing the burning as the size is stored in the root, which needs to be found anyway in order to find out whether the cluster is a border cluster) and the master can signal all other slices to send the border and to continue. After receiving the borders, it can update the histogram [56].

(iii) The site is occupied and contains a border site. In this case, the slice sends the reference of the border site back to the master, which then contacts all slices to send the most recent border update. It updates the border and the histogram, deletes the cluster which is going to burn, and sends the "burning borders," i.e., a list of all border sites which will be affected by the burning procedure to the slices in the form of a stack as described in Sec. II C 4. The slaves use this stack as the initial stack of the burning procedure and delete the corresponding sites. No communication between the slices is necessary.

The global histogram contains much larger clusters than the local histograms. In order to keep memory requirements low, even for histograms of resolution unity, it is reasonable to introduce a threshold above which slaves use the global histogram to maintain $\mathcal{P}^a$ even for local clusters (i.e., nonborder cluster). For that purpose, a histogram "appendix" has been introduced. This is a finite stack, which stores the size of the cluster s together with the value of $t'=\pm(T-t+1)$ as described in Sec. II C 3. During the growing phase when such large clusters grow fast, one would obtain a sequence of stack entries of the form (s,t'), $(s,-t')$, $(s+1,t')$, $(s+1,-t')$, $(s+2,t')$,..., corresponding to entering the appendix, (s,t'), increasing in size by 1, which gives $(s,-t')$, $(s+1,t')$, etc. As soon as a cluster is larger than the upper cutoff, each update causes two entries, of the form $(s,-t')$, $(s+1,t')$, the first for the deletion from the histogram, the second from the increase in the next slot. These entries possibly cancel, for example the sequence above is equivalent to the single entry $(s+2,t')$. It turned out to be highly efficient to perform this cancellation, i.e., to check the last entry in the appendix for being the negative entry of the one to be done.

As the maximum size of the appendix is finite, it must be emptied from time to time. The information about the size of the appendix of each slave is sent to the master together with the information about the borders. If a possible overflow is detected (2/3 of the maximum size in the implementation presented), the master requests all slices to send the content of their appendices and applies it to the global histogram. The slices then empty their appendices.

3. The random number generator

The random number generator (RNG) acquires a crucial role when used in a parallel environment. With M the number of iterations, the expected number of calls of the RNG is $M\theta^{-1}/\rho$ (for $M\approx 10^7$, $\theta^{-1}\approx 5\times 10^4$ this is more than 5×10^{11}), so that an RNG such as ran1 in [42] with a period of only $\approx 2\times 10^9$ is insufficient. Therefore, ran2 in [42] was used for all simulations, both parallel and nonparallel, which has a period of $>2\times 10^{18}$. If the number of RNG calls is small enough, one can compare results obtained by means of ran1 and ran2. No significant deviation was found.

In the parallel implementation, each slave requires an independent sequence of random numbers. This is a classical problem in parallel computing [43,44]. The simplest solution is to divide a single sequence r_1, r_2,... into distinct subsequences. This can be done either by a leapfrog scheme [44,45], where each subsequence consists of random numbers which are S calls away, i.e., S subsequences of the form r_u, r_{S+u}, r_{2S+u},... with $u=1,2,\ldots,S$ unique at each slave, or by splitting the sequence [44], so that each subsequence consists of consecutive RNG calls, i.e., r_{1+uX}, r_{2+uX}, r_{3+uX} again with $u=1,2,\ldots,S$ and offset X large enough to avoid any overlap. The latter scheme has the advantage that the sequence consists of consecutive RNG calls and therefore has been used in the following. The implementation of the offset X at each slave is easily realized by restoring all state variables of the RNG, which have been produced once and for all in a single run producing all XS random numbers and saving the state variables on a regular basis. However, such a technique is advisable only if the RNG calls do not dominate the overall CPU time, in which case it would take almost as long as the simulation itself to produce the random numbers required for it.

TABLE III. Parameters and results for different choices of L and θ^{-1}. The average cluster size is denoted by $\tilde{s}$, for definition see Eq. (1), but due to a truncation in the histogram for some of the simulations in the range $2000 \leq \theta^{-1} \leq 16\,000$, the number presented is actually the average size of the burnt cluster. In the stationary state it is—apart from small corrections—also given by $(1-\overline{\rho})/(\theta\overline{\rho})$, see Eq. (7). Values of θ^{-1} and L printed in bold indicate results shown in Fig. 16; the other results are only for comparison. All data are based on 5×10^6 (successful) updates (see Sec. II B 1) for the transient and statistics, apart from those printed in italics, which are based on short runs (5×10^6 updates for the transient and 1×10^6 updates for statistics).

θ^{-1}	L	$n(1)$	$\tilde{s}$	$\overline{\rho}$	$(1-\overline{\rho})/\theta\overline{\rho}$
125	*1000*	0.045 53	204.07	0.379 73	204.18
125	1000	0.045 52	203.81	0.379 77	204.15
125	*4000*	0.045 53	203.88	0.379 83	204.10
125	**4000**	0.045 52	203.77	0.379 83	204.10
250	*1000*	0.044 51	395.03	0.387 56	395.06
250	1000	0.044 52	394.08	0.387 50	395.15
250	*4000*	0.044 54	394.97	0.387 66	394.89
250	**4000**	0.044 54	395.29	0.387 65	394.91
500	*1000*	0.043 80	764.73	0.393 16	771.75
500	1000	0.043 80	764.81	0.393 15	771.77
500	*4000*	0.043 82	771.12	0.393 43	770.88
500	**4000**	0.043 82	771.90	0.393 43	770.87
1000	*1000*	0.043 28	1495.36	0.397 16	1517.91
1000	1000	0.043 28	1490.05	0.397 14	1518.00
1000	*4000*	0.043 31	1510.85	0.397 61	1515.00
1000	**4000**	0.043 31	1513.13	0.397 64	1514.81
1000	*8000*	0.043 32	1510.10	0.397 63	1514.91
2000	*4000*	0.042 96	2976.34	0.400 53	2993.35
2000	**4000**	0.042 97	2990.50	0.400 54	2993.15
2000	*8000*	0.042 97	2995.67	0.400 60	2992.56
4000	*4000*	0.042 73	5929.24	0.402 58	5935.91
4000	4000	0.042 73	5930.97	0.402 49	5938.03
4000	*8000*	0.042 74	5931.32	0.402 61	5935.15
4000	**8000**	0.042 73	5935.36	0.402 56	5936.47
8000	*4000*	0.042 55	11 786.97	0.404 05	11 799.72
8000	4000	0.042 55	11 788.90	0.404 06	11 799.07
8000	*8000*	0.042 57	11 801.31	0.404 12	11 795.98
8000	**8000**	0.042 57	11 792.82	0.404 13	11 795.38
16 000	*4000*	0.042 44	23 430.01	0.405 25	23 481.82
16 000	*8000*	0.042 43	23 466.93	0.405 40	23 467.22
16 000	8000	0.042 43	23 446.10	0.405 42	23 465.64
16 000	**16 000**	0.042 45	23 449.31	0.405 41	23 466.57
32 000	16 000	0.042 32	46 443.83	0.406 60	46 701.82
32 000	**32 000**	0.042 33	46 731.44	0.406 62	46 698.51
64 000	**32 000**	0.042 20	91 148.64	0.407 77	92 952.40

III. RESULTS

The sections above were only concerned with the technical issues of the model and its implementation. Some of the actual results from the simulation carried out using the new algorithm have been published already [18]. This article was focused on $\overline{n}(s)$. The main outcome was that the standard scaling assumption (12) is not supported by numerics, so the main conclusion was that the model *is not scale invariant*.

In the following, these results are shortly restated and discussed. Other observables are connected with this observation to see whether it is only $\overline{n}(s)$ which lacks scale invariance. All results presented are based on the same simulations, the parameters of which are given in Table III.

A. Cluster size distribution

Before the actual findings are discussed, it is important to consider how to avoid finite-size effects, which otherwise might damage the results. Usually, finite-size effects are avoided by keeping the correlation length ξ small compared to the system size. However, it requires a significant amount of CPU time to actually determine the correlation length.

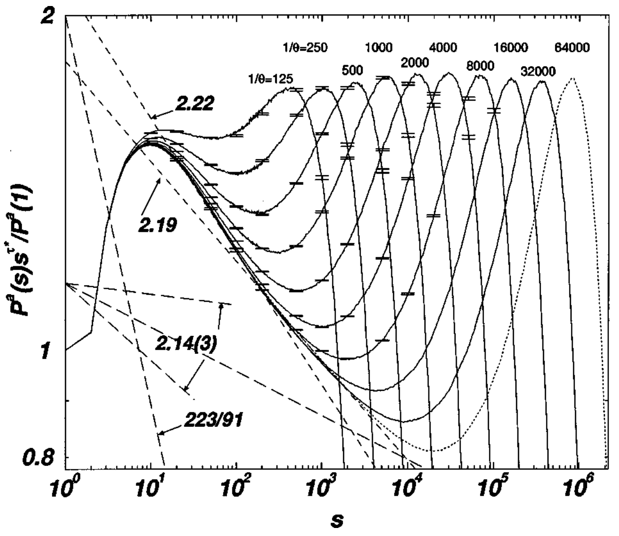

FIG. 16. The rescaled and binned histogram $\mathcal{P}^a(s)s^{\tau^*}/\mathcal{P}^a(1)$, where $\tau^*=2.10$ for $\theta^{-1}=125,250,500,\ldots,32\,000,64\,000$ (as indicated) in a double logarithmic plot. The linear size L is chosen according to the bold printed entries in Table III and large enough to ensure absence of finite-size effects. The error bars are estimated from shorter runs. The rightmost histogram (dotted, $\theta^{-1}=64\,000$) could not be cross-checked by another run, see text. The dashed lines belong to different exponents, whose value is specified as the sum of the slope in the diagram and τ^*, i.e., a horizontal line would correspond to an exponent 2.1. The short-dashed lines represent estimated exponents for different regions of the histogram (2.22 for s within approximately [20,200] and 2.19 for s within [200,2000]); the other exponents are from the literature, namely 2.14(3) in [15,24] and 223/91≈2.45 in [46]. Since it was impossible to relate these exponents to any property of the data, the exact position of the lines associated with them was chosen arbitrarily.

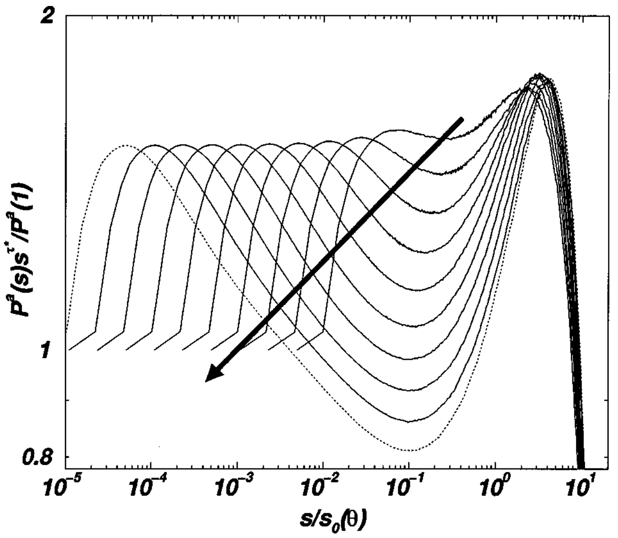

FIG. 17. Attempt to collapse the data shown in Fig. 15 using $\tau^*=2.10$, $s_0(\theta)=\theta^{-\lambda^*}$, and $\lambda^*=1.11$ as derived from Eq. (21). As expected, the data do not collapse. The big arrow points in the direction of increasing θ^{-1}.

Moreover, *a priori* it would not be clear which ratio ξ/L to choose in order to avoid finite-size effects.

The simplest way to determine whether finite-site effects are present is to compare estimates of observables for two systems with the same parameters but different sizes [40]. If finite-size effects are not present, the differences between the estimators of those two systems must be within the error bar of the quantity under consideration. This approach has the drawback that each set of parameters must be simulated at least twice, but it gives full control over finite-size effects. Apart from $\theta^{-1}=64\,000$, which is specially marked in most of the plots, this approach has been applied throughout the results presented. The method was discussed in greater detail in [18].

Figure 16 shows a central result of [18]. This figure contains the reduced (and binned) data in the form

$$\frac{\mathcal{P}^a(s)}{\mathcal{P}^a(1)}, \tag{41}$$

which has the convenient property to be unity for $s=1$. The normalization $\mathcal{P}^a(1)$, which converges anyway to a finite value as $\theta^{-1}\to\infty$ (see Table III), does not affect any of the results, especially not the (attempted) data collapses.

The crucial problem shown in Fig. 16 is the intermediate minimum that develops as θ^{-1} is increased. It renders the data collapse as described by Eq. (12) impossible (for more details, see [18]). Figure 17 shows the same data again, now in an attempt to form a data collapse, using $s_0(\theta)=\theta^{-\lambda^*}$ with $\lambda^*=1.11$ from Eq. (21) and $\tau^*=2.10$ (for comparison, see Table IV). As expected, the collapse fails.

In less technical terms, it was shown in [18] that there is no choice of τ, which allows a data collapse for $\mathcal{P}^a(s;\theta)$. It seems that the distribution is the same for two different values of θ up to a certain cluster size, which increases seemingly unbound with θ^{-1}, i.e., for two very large values of θ^{-1} the two distributions collapse without any rescaling. Beyond this cluster size, the distributions deviate. The one with the larger θ^{-1} forms a deeper dip and ascends afterwards to a maximum, which can, by rescaling, be arranged to be the same for all θ^{-1}. The ever growing dip prohibits a reasonable definition of a lower cutoff and makes a data collapse impossible. Equally one could arrange the dips to be at the same height and the maximum to increase in θ^{-1}.

The key problem of the DS-FFM is that more than one length scale is visible apparently for any system size L. The statistics of $\overline{n}(s)$ is not even asymptotically dominated by a single length scale. For any system size, an $\overline{n}(s)$ only given for all s larger than any lower cutoff allows the identification of θ by the shape of $\overline{n}(s)$ alone.

This indicates that simple scaling (12) does not apply and the exponent τ is undefined. Keeping this in mind, it is very instructive to look for other properties as well and investigate their scaling.

1. Finite-size scaling

The failure of the DS-FFM to obey proper finite-size scaling has been observed in [22] already. In the following, some finite-size scaling principles have been applied in a straightforward manner and subsequently ruled out.

As known from percolation [14], the generalized form of the scaling behavior of s_0 is

$$s_0(\theta,L)=\theta^{-\lambda}m(\theta L^{\sigma}), \tag{42}$$

where $m(x)$ is a crossover function describing the dependence of s_0 on the two parameters θ and L. For sufficiently large argument x, the crossover function is expected to approach a constant, such that Eq. (20) is recovered. For small arguments, however, the dependence of the cutoff is expected to be strongly dominated by L, just like in equilibrium critical phenomena, where L takes over the role of ξ for sufficiently small systems. Thus, for small arguments $m(x) \propto x^{\lambda}$, so that for sufficiently small L, s_0 becomes independent of θ.

Generic models of SOC do not have any tuning parameters other than the system size, so that the cutoff s_0 is only a function of L. In this sense, finite-size scaling is the only scaling behavior in SOC, and a failure of the model to comply to finite-size scaling is identical to the failure to comply to simple scaling altogether. Therefore, one might be surprised to see a simple scaling analysis *and* a finite-size scaling analysis in an article on an SOC model. However, the forest fire model is different in this respect, as it has the additional parameter θ, which is, supposedly, finite only because of the finiteness of the system size. In the thermodynamic limit, it supposedly disappears as a free parameter.

As seen above (see Fig. 17), the θ dependence of $\overline{n}(s;\theta)$ cannot be captured by s_0 in the scaling function alone. However, the scaling form (12) would remain valid in some sense if in the finite-size scaling regime the L dependence of $\overline{n}(s;\theta)$ enters s_0 only. Therefore, the original form (12) is generalized to

$$\overline{n}(s;\theta,L)=s^{-\tau}\mathcal{G}(s/s_0(\theta/L)), \tag{43}$$

ignoring that it has been shown above already that it does not hold in the limit where $\overline{n}(s;\theta,L)$ becomes independent of L. In this section, the dependence of $\overline{n}(s;\theta,L)$ on L is investigated in the limit of large θ^{-1} and small L. A similar study has been performed by Schenk *et al.* [22], however on much smaller scales and using $\mathcal{P}^b$.

If the form (43) holds, it should be possible to collapse $\overline{n}(s;\theta,L)$ for different L by choosing the correct τ and s_0, just like for the cluster size distribution of standard percolation. This turns out not to be the case, as can be seen in Fig. 18: The *smaller* L is, the *stronger* the changes of shape of $\overline{n}(s)$ for any θ tested. Consequently, Eq. (43) does not hold, and as s_0 is only *defined* via its role as cutoff in Eq. (43), s_0 is undefined and Eq. (42) remains meaningless.

One might argue that the average density of trees, $\overline{\rho}$ [see Eq. (4)], is the relevant parameter of $\overline{n}(s)$, so that $\overline{n}(s)$ has the same shape for different, sufficiently small L and constant $\overline{\rho}$. However, as shown in Fig. 20, for any value of θ there is a value of L, such that $\overline{\rho}$ varies considerably with decreasing L. Especially, there seems to be a maximum tree density for every system size, so that for large values of $\overline{\rho}$ there is a smallest system size L, below which this density cannot be reached. This maximum increases monotonically

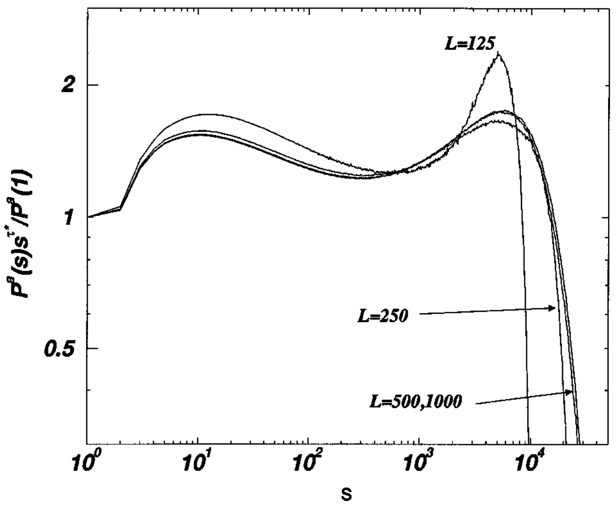

FIG. 18. Plot of the rescaled PDF $\mathcal{P}^a(s;\theta,L)s^{\tau^*}/\mathcal{P}^a(1;\theta,L)$ for fixed $\theta^{-1}=1000$ and different system sizes, $L=125,250,500,1000$. The different shapes make it impossible to collapse the data, as would be expected from a finite-size scaling ansatz (43) and (42).

with system size, so that the maximum for every finite system size is smaller than the expected average tree density in the thermodynamic limit, which is, according to Table III, larger than 0.407 77 and was recently conjectured to be as large as 0.5927 . . . [47], namely the critical density of site percolation [29]. Accepting this limitation, Fig. 19 shows an example for three $\overline{n}(s)$ with roughly the same $\overline{\rho}$ and different L and θ. Most surprisingly, two of the histograms collapse already without rescaling, while the third ($L=500$) reveals the same problems as visible in Fig. 16. Hence, finite-size scaling also does not work for fixed $\overline{\rho}$.

That large densities of trees cannot be reached by small system sizes is related to the specific way the histograms are generated and the density measured: Is it before or after each

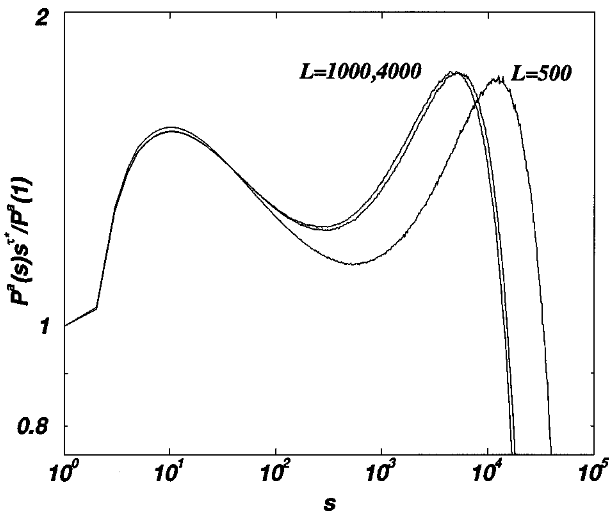

FIG. 19. Plot of the rescaled PDF $\mathcal{P}^a(s;\theta,L)s^{\tau^*}/\mathcal{P}^a(1;\theta,L)$ for fixed $\overline{\rho}\approx 0.397$: $L=500$ with $1/\theta=2000$ ($\overline{\rho}=0.396\,827$), $L=1000$ with $1/\theta=940$ ($\overline{\rho}=0.396\,825$), and $L=4000$ with $1/\theta=870$ ($\overline{\rho}=0.396\,883$). Again, a data collapse is impossible.

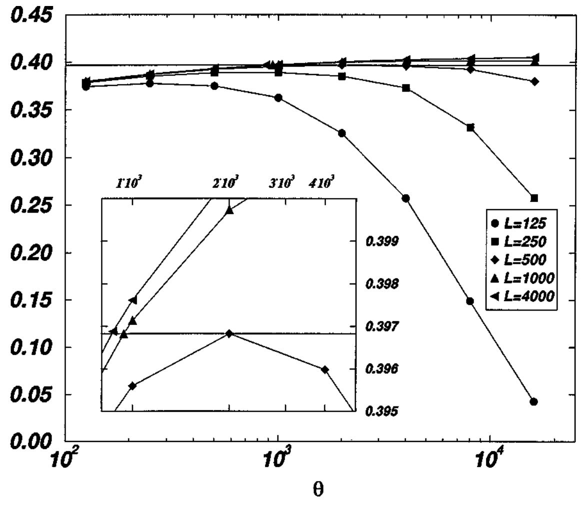

FIG. 20. The average density of trees, $\overline{\rho}$, as a function of θ and for various L. For sufficiently small systems, the maximum in $\overline{\rho}$ is much smaller than the expected density at the "critical point," which is larger than 0.407 77 found as in Table III. The straight line marks $\rho = 0.396\,827$, the density chosen in Fig. 19. The inset is a magnification of the crossing of the straight line with the simulation data, and shows all three values of θ, L used in Fig. 19.

(successful) burning? For sufficiently large systems, it becomes irrelevant when to do it, because two histograms, one measured before, the other one right after the burning, differ only by one cluster. Also the question of whether to average only over successful burnings is irrelevant, because the difference between a histogram before and after the burning is only one cluster.

Clearly, for small systems, the difference between the histogram before and after the burning is just the one enormous cluster of size $\mathcal{O}(\theta^{-1})$. Figure 21 shows the difference. Even though in principle every density is reachable for every system size if the histogram is measured before burning, the newly defined histograms do not have a considerably different shape, so that a collapse remains impossible. For example, the problems shown in Fig. 18 become even more pronounced, if the histogram is taken before burning.

Surprisingly and actually in contradiction to what has been said in Eq. (6), there is a discrepancy between the cluster size distribution of burnt clusters, $\mathcal{P}^b$, and the overall cluster size distribution $\mathcal{P}^a$, even if the latter is measured *before* the burning takes place. This sounds paradoxic, because the random picking of a cluster to be burnt is just a sampling of $\mathcal{P}^a$. This cannot be caused by the correlation between those samples, due to the fact that $n_{t+1}(s)$ is actually a function of the cluster chosen at t—a correlation like this would be equally picked up by $\mathcal{P}^a$. The reason for this discrepancy is the fact that a site picked randomly as the starting point of the next fire is necessarily occupied. Therefore, $n_t(s)$ with a low occupation density enters $\mathcal{P}^b$ overweightedly. As low density states contain many more small clusters than large ones, $\mathcal{P}^b$ overestimates the probability of small clusters. A sample for $\mathcal{P}^b$ at a low density is indistinguishable from a sample at high density, while a sample for $\mathcal{P}^a$ trivially contains the information about the density. To illustrate that, one might imagine a sequence of (burnt) configurations that consists of one state, with exactly one cluster of size 1, and a second state, with exactly one cluster of size L^2. The two configurations appear with a frequency such that a cluster of size 1 is burnt down as often as a cluster of size L^2. The resulting $\mathcal{P}^a$ reports that a randomly chosen site belongs to a cluster of size L^2 with probability $\frac{1}{2}$ and to a cluster of size 1 with probability $1/(2L^2)$, while $\mathcal{P}^b$ incorrectly reports the same probability for both cluster sizes. The problem can actually already be spotted in Eq. (6), which contains a ρ on the RHS, which should rather be $\rho(t)$. The problem disappears in the limit where $\rho(t)$ hardly changes in time, i.e., in the limit of $\theta^{-1} \ll L^2$.

It is also clear why Eq. (7) breaks down for small systems and large θ^{-1}: The average size of the burnt cluster tends to L^2, while the density tends to 0. Apparently Eq. (7) must be incorrect for $\rho < (L^2\theta + 1)^{-1}$.

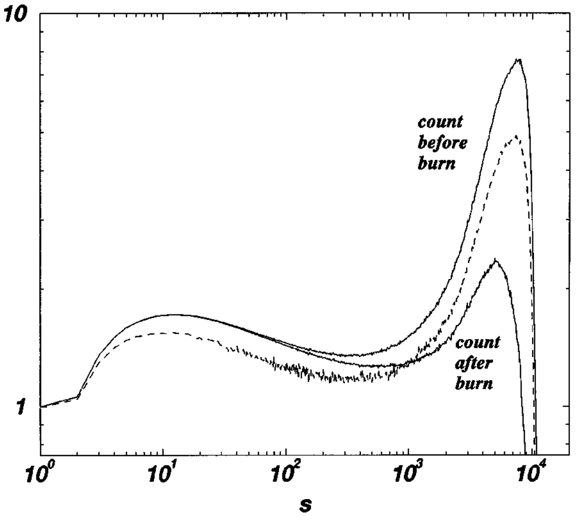

FIG. 21. Comparison between the rescaled and binned histograms measured before and after the burning for small $L = 125$ and large $\theta^{-1} = 1000$. As expected, only the statistics for large s is affected. The dashed line shows the data for $\mathcal{P}^b(s)$.

2. Scaling of the moments of $\mathcal{P}^a$

According to Eqs. (12), (20), and (8), the nth moment of $\mathcal{P}^a$ should scale like (this analysis has apparently been introduced to SOC by De Menech *et al.* [25,48,49])

$$\widetilde{s^n} = \frac{\Sigma_s s^n s\overline{n}(s;\theta)}{\Sigma_s s\overline{n}(s;\theta)} = q_n \theta^{-\lambda(2+n-\tau)} + \text{corrections}, \quad (44)$$

where q_n is a nonuniversal amplitude (see Sec. III A 3) and λ is also known as a gap exponent [50]. The corrections are due to the lower cutoff and the asymptotic character of the scaling, which is expected only for "sufficiently small θ" [26]. In turn, one can infer a scaling form like Eq. (12) if the moments scale in the form of Eq. (44).

Contrary to what is observed in an attempt of a data collapse, it turns out that the moments follow beautifully this scaling behavior. Figure 22 shows the scaling of the mo-

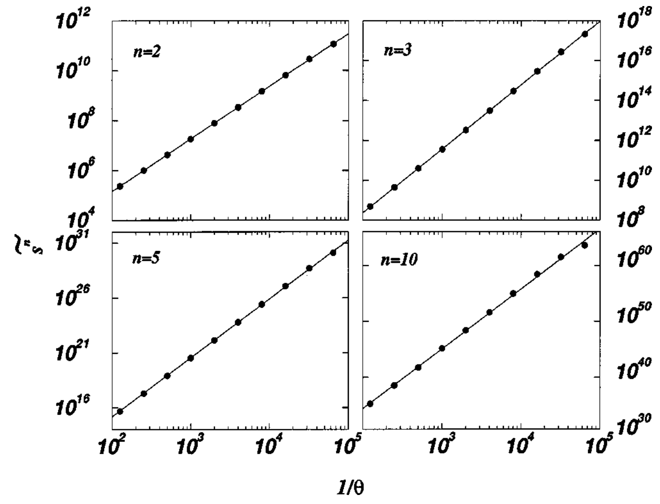

FIG. 22. Scaling of the nth moments of $\mathcal{P}^a$ in double logarithmic plots. The straight lines show the results of a fit as $\exp(a_n')\theta^{-\sigma_n}$, see Eq. (44).

ments for $n=2,3,5,10$. By simply fitting the double logarithmic data to a straight line, i.e.,

$$\log(\widetilde{s^n}) = a_n' - \sigma_n \log(\theta), \tag{45}$$

one can derive an estimate of the exponents σ_n and in turn compare them to the expected linear behavior,

$$\sigma_n = \lambda(2+n-\tau). \tag{46}$$

The resulting estimates, using $n=2,\ldots,8$ and $\sigma_1=1$ from Eq. (1), are $\lambda=1.0808\ldots$ and $\tau=2.0506\ldots$, where no statistical error is given because the systematic error, due to neglecting of the lower cutoff as well as the corrections (44), is expected to be much more important. By using the assumption $\sigma_1=1$, this result is consistent with Eq. (21). The results are shown in Fig. 23.

The exponent found for τ is remarkably close to the accepted value of standard 2D percolation, $187/91 = 2.054\,945\ldots$. However, if one leaves out the results for $\theta^{-1}=64\,000$, which seem to be a bit off the lines shown in Fig. 22, one finds a slightly larger value for the exponent, namely $\tau=2.0864$ and $\lambda=1.0998\ldots$. This is much closer to the $\tau^*=2.10$ used above. For comparison to values found in the literature, see Table IV.

It is very remarkable that the resulting estimates for the exponents are so impressingly consistent, even though in Sec. III A it turned out that the scaling assumption (12) does not actually hold; one would much rather expect a failure of the moments to comply with Eq. (44), or a failure of the exponents to comply with Eq. (46). Apparently the moments are hiding the breakdown of simple scaling. Therefore, it is interesting to analyze the behavior of the presumably universal amplitude ratios, which are solely a property of the (presumed) scaling function.

Another explanation for the moments being well behaved is the following: According to [18], one might expect the moments to behave like

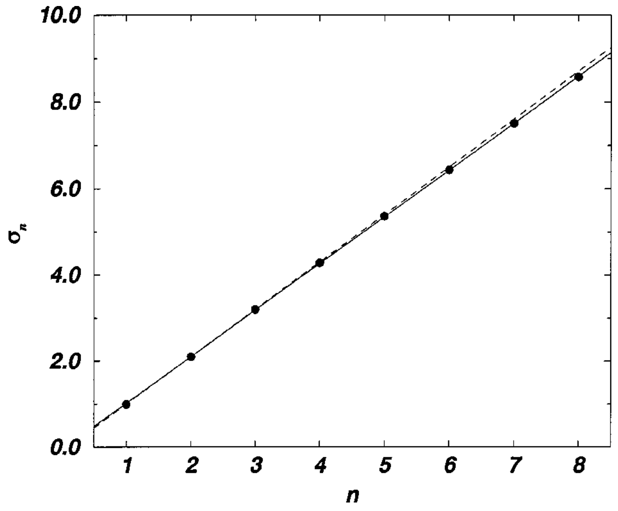

FIG. 23. Exponents σ_n of the scaling of $\widetilde{s^n}$ in θ vs n. The slope of this curve gives λ, and τ can be derived from the offset. The straight, full line shows the results $\lambda=1.0808\ldots$ and $\tau=2.0506\ldots$; the dashed line shows $\lambda=1.0998\ldots$ and $\tau=2.0864\ldots$ from a fit excluding $\theta^{-1}=64\,000$.

$$\int_1^{\theta^{-x_{\min}}} ds\, f(s)s^n + \int_{\theta^{-x_{\min}}}^{\infty} ds\, s^{n-\tau}\mathcal{G}(s/\theta^{-x_{\max}}), \tag{47}$$

where the first integral describes the behavior up to the minimum, which scales like $\theta^{-x_{\min}}$ ($x_{\min}\approx 0.95$), and the second integral the behavior from the minimum on. Because Fig. 17 indicates already that the scaling function $\mathcal{G}$ does not collapse using a scale $\theta^{-x_{\max}}$, this scaling does not work and can therefore be only an approximation. While the first integral is bound by $O(\theta^{-(n+1)x_{\min}})$, the second integral gives $O(\theta^{-(n+1-\tau)x_{\max}})$ asymptotically, which dominates the moments for $x_{\min}(n+1) < x_{\max}(1+n-\tau)$, which leads to $n > 9.08$ using $x_{\max}\approx 1.2$ and $\tau\approx 2.1$. Figure 22 shows clearly a deviation from the straight line behavior for $\theta^{-1}=64\,000$ and $n=10$ and even for $n=5$. It remains unclear whether this is due to the effect discussed or simply a finite-size problem. According to the findings presented in Sec. III A 3, the latter might well be the case.

It is worthwhile to point out that the analysis in this section arrives at estimates for the critical exponents very close to those obtained by Pastor-Satorras and Vespignani [25], who, however, allow for the corrections in Eq. (44) that were omitted above.

3. Universal amplitude ratios

In general, simple scaling involves two additional nonuniversal parameters a and b,

$$\overline{n}(s;\theta) = a s^{-\tau}\mathcal{G}\left(\frac{s}{b s_0}\right). \tag{48}$$

For $1<\tau<2$, the lower cutoff becomes asymptotically irrelevant compared to the upper cutoff for all moments $n\geq 1$—indeed the effective τ of $s\overline{n}(s;\theta)$ fulfills this condi-

TABLE IV. Exponents of the forest fire model found in the literature. The first column indicates the source, the second column the method. $P(s)$ denotes a direct analysis of $\overline{n}(s;\theta)$, which sometimes may have been just an estimate of the slope of $\overline{n}(s;\theta)$ rather than a data collapse. For details, the original sources should be consulted. The entry "moments" refers to an analysis of the moments of $P(s)$, the entry "theoretical" to theoretical considerations regarding the relation of the forest fire model to percolation.

Reference	Method	τ	λ
Christensen *et al.* [16]	$P(s)$	2.16(5)	
Henley [21]	$P(s)$	2.150(5)	1.167(15)
Grassberger [17]	$P(s)$	2.15(2)	1.08(2)
Clar *et al.* [15]	$P(s)$	2.14(3)	1.15(3)
Honecker and Peschel [20]	$P(s)$	2.159(6)	1.17(2)
Pastor-Satorras Vespignani [25]	moments	2.08(1)	1.09(1)
Schenk *et al.* [46]	theoretical and $P(s)$	2.45 . . .	1.1
Grassberger [47]	$P(s)$	2.11	1.08

tion as $2<\tau<3$ [24]. Neglecting the lower cutoff then gives for the nth moment of $s\overline{n}(s;\theta)$

$$\widetilde{s^n}=a(bs_0)^{1+n-\tau}g_n \tag{49}$$

with

$$g_n \equiv \int_0^\infty dx\, dx^{1+n-\tau}\mathcal{G}(x). \tag{50}$$

In order to construct universal amplitude ratios, one needs to get rid of all exponents and parameters. This can be achieved by considering

$$\frac{\widetilde{s^n}}{(\widetilde{s^2})^{n/2}}=[a(b\theta^{-\lambda})^{(1-\tau)}]^{(1-n/2)}\frac{g_n}{g_2^{n/2}} \tag{51}$$

and [Eq. (51) with $n=1$]

$$\frac{\tilde{s}}{\sqrt{\widetilde{s^2}}}=[a(b\theta^{-\lambda})^{(1-\tau)}]^{1/2}\frac{g_1}{g_2^{1/2}}. \tag{52}$$

If one now multiples Eq. (51) with the $(n-2)$th power of Eq. (52), everything cancels apart from the g_n,

$$\frac{\widetilde{s^n}}{(\widetilde{s^2})^{n/2}}\,\frac{(\tilde{s})^{(n-2)}}{(\widetilde{s^2})^{(n-2)/2}}=\frac{g_n g_1^{n-2}}{g_2^{n-1}}. \tag{53}$$

It is worth noting that for a trivial case, where $\widetilde{s^n}\propto(\tilde{s})^n$, the effective exponent τ is necessarily unity, and Eq. (51) as well as Eq. (52) are already independent of θ.

A further simplification is to impose $g_1=1$ and $g_2=1$, which fixes the two free parameters a and b in Eq. (48), so that

$$g_n=\frac{\widetilde{s^n}(\tilde{s})^{(n-2)}}{(\widetilde{s^2})^{(n-1)}} \tag{54}$$

for $n\geqslant 1$. In Fig. 24, this quantity is shown for $n=3,4,5,6$. Now, for $\theta^{-1}=64\,000$ a deviation is clearly visible—in turn that means that $\theta^{-1}=64\,000$ requires at least systems of the size $L=64\,000$, which might explain the large value of $\overline{\rho}$ obtained in [47]. Apart from that, this analysis agrees with the result found in Sec. III A: The supposedly universal amplitude ratios keep changing with θ and an asymptote cannot be estimated, i.e., the scaling (12) is broken.

4. Burning time distribution

Another distribution of interest is the distribution of burning times, $\mathsf{P}_{T_M}(T_M;\theta)$. The statistics are comparatively small for this quantity, as the burning time is defined only for the cluster removed. However, they still seem to be good enough to allow us to make a statement about their scaling behavior. The rescaled data $\mathsf{P}_{T_M}(T_M;\theta)T_M^{b^*}$ with a trial exponent $b^*=1.24$ can be seen in Fig. 25. The intermediate part of the distribution between $T_M=4$ and the maximum seems to bend down as θ^{-1} increases, but the developing dip is much less pronounced than in Fig. 16. Nevertheless, the region where a data collapse seems possible moves out towards larger values of T_M, which again prohibits simple

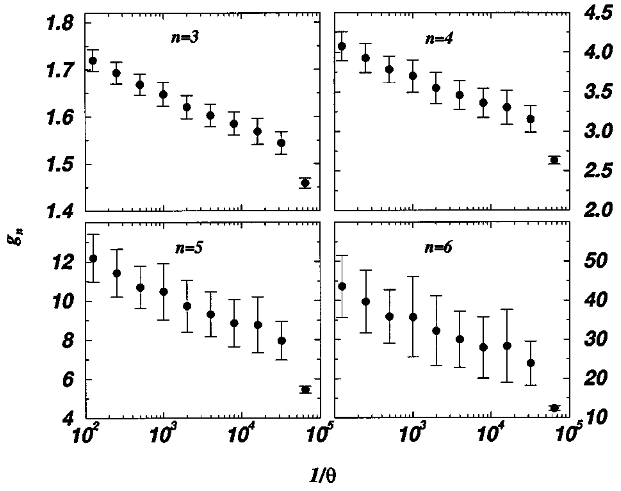

FIG. 24. The supposedly universal amplitude ratio g_n (54) for $n=3,4,5,6$. The error bars are based on a jackknife scheme [39,40] using a roughly estimated correlation time of 50, see Table II.

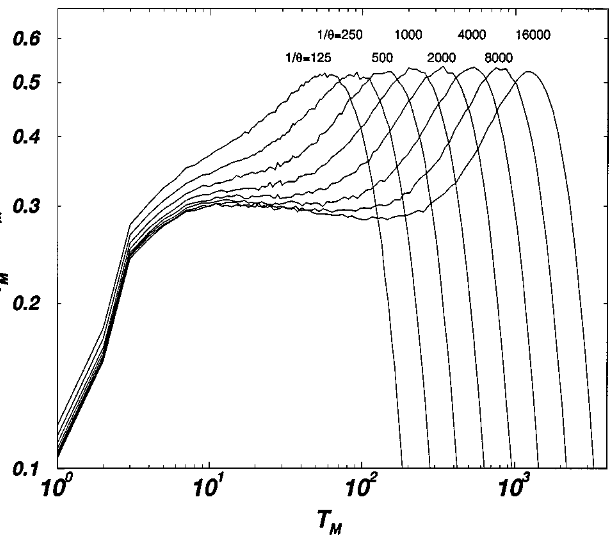

FIG. 25. The rescaled probability distribution of the burning time, $\mathsf{P}_{T_M}(T_M;\theta)$. Similar to Fig. 16, a dip seems to form between the low T_M region and the maximum, which again renders a data collapse impossible.

scaling. Assuming that the bending might become weaker for sufficiently large T_M leads to a data collapse shown in Fig. 26, using an exponent $\nu'=0.6$ as defined in Eq. (22). However, only for values of $T_M \approx T_{M_0}$ will the data possibly collapse. Again, this violates the assumption of simple scaling, namely that there is a *constant* lower cutoff above which the behavior is universal.

The only remaining exponent of those defined in Sec. II B 4, μ', relates the statistics of s and T_M. It requires the bivariate distribution $\mathsf{P}(s,T_M;\theta)$, as the exponent is derived from $\mathsf{E}(T_M|s)\propto s^{1/\mu'}$, which is essentially equivalent to Eq. (17). The distribution $\mathsf{P}(s,T_M;\theta)$ is shown in Fig. 27. At

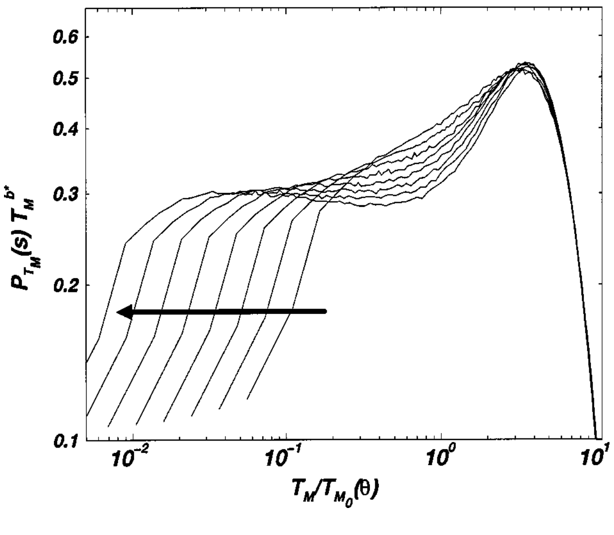

FIG. 26. Attempt of a data collapse for $\mathsf{P}_{T_M}(T_M;\theta)$. Only at the far end of the scaling function at the descent from the maximum do the data seem actually to collapse. This, however, is not sufficient for a data collapse. The big arrow points in the direction of increasing θ^{-1}.

first glance, the assumption of a power-law dependence of s and T_M seems to be confirmed. Also the width of the distribution seems to be very small, with almost no change over five orders of magnitude in s. However, the plot is double logarithmic, so that the width roughly scales like the slope, which is about 0.6, as shown by straight lines. This matches perfectly the exponent chosen to rescale P (see the caption of Fig. 27).

By inspecting $\mathsf{E}(T_M|s;\theta)$ and $\mathsf{E}(s|T_M;\theta)$ for various θ, one can determine μ' as a slope in a double logarithmic plot. Figure 28 shows that μ' remains ambiguous and deviations from the expected behavior do not vanish as θ^{-1} is increased. Asymptotically one might expect $1/\mu'\approx 0.62$, while $(\tau^*-2)/(b^*-1)$ suggests $1/\mu'\approx 0.417$. The value of 0.62 is consistent with the rough estimate 0.6 made in Fig. 27. Figure 28 also shows two other exponents, 0.53 and 0.7, the former being in line with the value found in the literature of 0.529(8) [15].

Conclusively, it is noted that the other observable available in this study, T_M, does not seem to provide an alternative way to ascribe the DS-FFM critical behavior in the sense of the scaling behavior as proposed in the literature.

B. Tree density as a function of time

As mentioned above (see Sec. III A 1), the density of trees, $\overline{\rho}$, is actually a function of time. Initially, it is periodic around the average value, with an amplitude that depends mainly on θ. This amplitude decays in time and after sufficiently long times $\rho(t)$ looks like a random walk around $\overline{\rho}$.

Figure 29 illustrates how the period and the amplitude depend on θ and L: The period is proportional to θL^2, while the amplitude mainly depends on θ, i.e., the strength of the influx $\propto \theta^{-1}$. The reason for the former is easy to understand: θ^{-1}/L^2 is proportional to the fraction of newly grown trees [20]; the change of the tree density is roughly

$$\frac{d}{dt}\rho = \frac{1-\rho}{\rho}\frac{1}{\theta L^2} - \eta(\rho(t),t) \tag{55}$$

assuming that it hardly changes during the growing. Otherwise, one would have to introduce a microscopic time scale, which makes it possible to measure the tree density on the time scale on which the trees are grown. The prefactor $(1-\rho)/\rho$ takes into account that only empty sites can be reoccupied and that an occupied site is required for the burning to start. The second term on the right-hand side, $\eta(\rho(t),t)$, is a noise which represents the burning of the trees. From this equation, one can already expect that the period is roughly linear in $\theta L^2\overline{\rho}/(1-\overline{\rho})$. This has already been measured in detail by Honecker and Peschel [20]; the numerical results presented here (Fig. 29) are fully consistent with their results.

Apart from the relevance of the periodic behavior for the equilibration time, the periodic behavior of $\rho(t)$ is physically of great significance: What distinguishes the state of the system for a given ρ at the ascending and the descending branches? Trivially, the sequence of configurations of the system in time is Markovian, while the tree density alone as

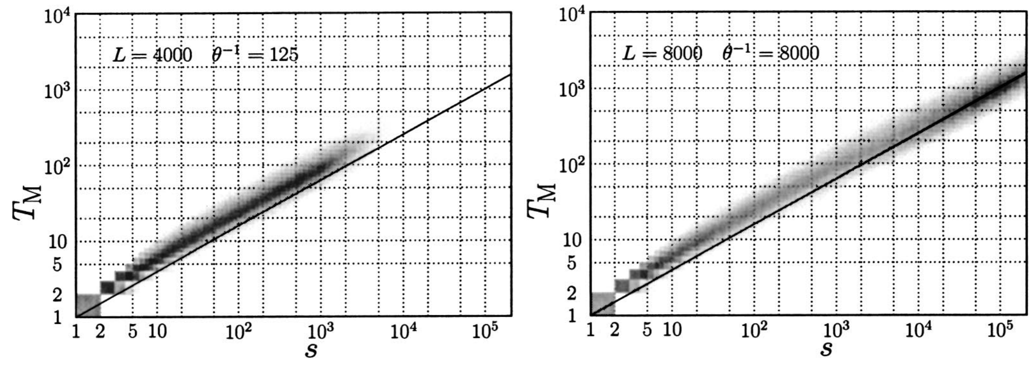

FIG. 27. Binned density plots of $\mathsf{P}(s,T_M;\theta)$ for different values of θ on a double logarithmic scale. High densities are presented as dark areas. For better presentation, $\mathsf{P}(s,T_M;\theta)$ has been multiplied by a factor $s^{1.7}$, tilting the distribution similar to those shown in Fig. 16, so that the second maxima in the distribution, those at large s and T_M, are roughly as high as the first maxima, i.e., they show in the plot as dark as around $s=5$. Since $\mathsf{P}(s,T_M;\theta)$ is a histogram only of burnt clusters, it contains a factor s compared to $\overline{n}(s)$ [see the discussion around Eq. (4)]. Therefore, the exponent 2.7 needs to be compared to $\tau^*=2.10$, indicating that the width of $\mathsf{P}(s,T_M;\theta)$ roughly scales like $s^{0.6}$, so that the reduced height of $\mathsf{P}(s,T_M;\theta)$ is caused by an increase in width. This coincides well with the slope of the distribution, as shown by a straight line. Thus, the relative width remains roughly constant.-

a time series is certainly not. The configuration somehow manages to "remember" whether the tree density was increasing or decreasing during the last update, in order to keep $\rho(t)$ periodic.

One explanation for this behavior might be a "growing-and-harvesting" concept: From the initially completely random tree distribution, larger and larger patches are formed, so that larger and larger patches are harvested by lightning. When the density reaches the maximum, for a while the patches harvested remain large compared to the amount grown. This is because the growing process does not actually produce those large patches itself, but makes them available to the harvesting by continuously connecting smaller patches in areas where the lightning has not yet struck. This process goes on until almost all the trees are newly grown, i.e., the trees are distributed almost randomly, apart from the spatial correlation in density. The period of this process would be proportional to the time it takes to renew the entire system, which is $L^2\theta\overline{\rho}/(1-\overline{\rho})$, namely L^2 divided by $\tilde{s}$, see Eq. (7).

The time-dependent tree density gives only a hint of what actually happens in the system. It would be very instructive to study the two-point correlation function as a function of time to answer the question of whether the explanation above is actually valid.

C. Discussion

From the results presented above, it becomes clear that the forest fire model does not show the scaling behavior expected for a system, which becomes critical in the appropri-

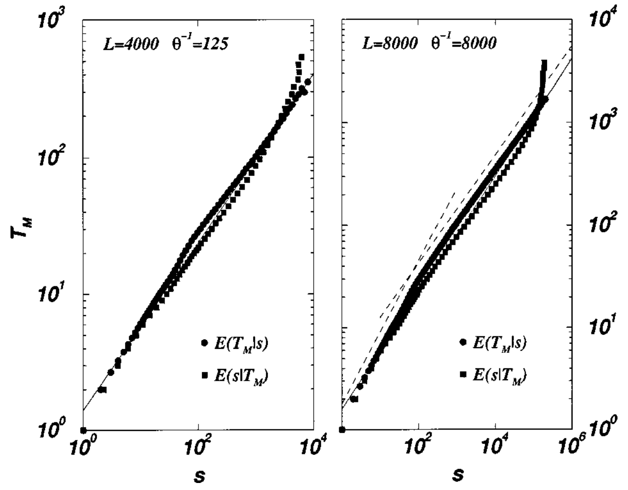

FIG. 28. $\mathsf{E}(T_M|s;\theta)$ and $\mathsf{E}(T_M|s;\theta)$, based on the binned histogram $\mathsf{P}(s,T_M;\theta)$ for different values of θ^{-1}. The straight lines in the plots are $1.4s^{0.615}$ for $\theta^{-1}=125$ (left-hand plot) and $1.6s^{0.57}$ for $\theta^{-1}=8000$. The two dashed lines in the right-hand plot show alternative exponents $1/\mu'=0.7$ and $1/\mu'=0.53$, which are consistent with data for small values of s or for large values.

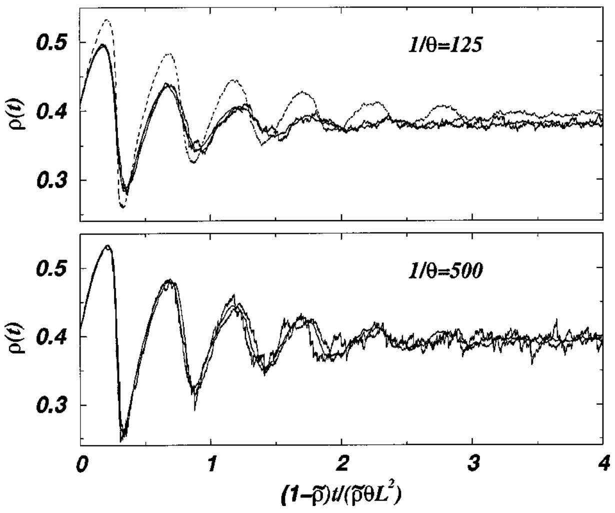

FIG. 29. The density of trees as a function of time, plotted versus the rescaled time $(1-\overline{\rho})t/(\theta\overline{\rho}L^2)$. Upper panel: Plot for $\theta^{-1}=125$ and $L=1000,2000,4000$ with an additional plot for $\theta^{-1}=500$ and $L=4000$ shown as a dashed line, for comparison of period and amplitude. Lower panel: Same plot for $\theta^{-1}=500$ and $L=1000,2000,4000$.

ate limit (namely $L \rightarrow \infty$ and $\theta^{-1} \rightarrow \infty$). One might argue that another scaling ansatz could lead to a distribution which is asymptotically scale-free in this limit, for example a multifractal ansatz [49] or the one proposed in [46], where more than one scale is assumed to govern the model. For an asymptotically scale-free distribution, the scales have to diverge or to vanish in the appropriate limit. It has been suggested already very early [20] that more than one characteristic length scale can be found in the forest fire model.

However, changing the scaling assumption would entail a new *definition* of the exponents τ, D, etc., which would therefore prohibit comparison with other results which are based on the assumption of simple scaling (12). Moreover, introducing multiple scales would stretch the notion of universality, especially the universality of the scaling function, to its limits. As can be seen in Fig. 16, the shape of the distribution function *is not universal*, i.e., the shape of this function is unique for every single θ^{-1}, even for $L \rightarrow \infty$. This is in direct contradiction to the concept of universality, scaling, and scale invariance.

However, it might be possible to reestablish simple scaling by introducing another mechanism in the model, as was done, for example, in the "autoignition forest fire model" [51]. If there were, for example, a mechanism parametrized by u, such that

$$\overline{n}(s;\theta,u) = s^{\tau}\mathcal{G}(s/s_0(\theta,u)), \tag{56}$$

then simple scaling might be reestablished possibly by choosing an appropriate $u = u(\theta)$; even the cutoff s_0, which was assumed to diverge with θ^{-1}, would then effectively depend only on θ. Currently, there is no hint of what this new parameter u could be.

Lise and Paczuski [8] suggested for a similar problem in the OFC model [7] to define an exponent τ by the slope of the distribution $\mathcal{P}^a(s)$, imposing the remaining background, $\mathcal{F}(s,L,\theta^{-1})$, to be as straight as possible,

$$\ln(\mathcal{P}^a(s)) = -\tau \ln(s) + \mathcal{F}(s,L,\theta^{-1}). \tag{57}$$

This ansatz, in fact based on a multiscaling ansatz, would indeed allow the measurement of an exponent, however with some degree of ambiguity. The crucial problem with this approach is that, first, it again does not allow any direct comparison to other models, where the exponents are defined via Eq. (12) and that, secondly, the notion of a presumably universal exponent hides the fact of broken scaling.

From Sec. III A, one might conclude that there does not even exist a limiting distribution for $\overline{n}(s;\theta)$. However, even if it exists, that does not mean that simple scaling is obeyed, and if it does, it is still open whether the exponents are nontrivial or not and whether the model posseses any spatiotemporal correlation which does not vanish on sufficiently large scales.

IV. SUMMARY

Using an alternative method for simulating the forest fire model on large scales, it is possible to make clear statements about the validity of the scaling assumption of this model. The two observables investigated in this paper suggest the model does not develop into a scale invariant state.

The method is based on the Hoshen-Kopelman algorithm [32] and uses a master-slave parallelization scheme to simulate the model on very large scales and very large sample sizes. The key to the parallelization is to decompose the lattice in strips and to encode the connectivity of these strips in the border sites. Clusters crossing these strips are then maintained by the master node, while clusters within a strip are maintained on the local nodes. There is almost no data exchange apart from the border configuration, which lowers the impact on the network linking the nodes.

The resulting distribution $\mathcal{P}^a(s)$ is, unlike other simulations found in the literature, the distribution of *all* clusters in the system, rather than just the burnt clusters. The resulting statistics then allows us to draw clear conclusions as to what extent the model does actually obey the scaling assumption. This turns out not to be case. The violation of scaling is also observed in the distribution of the burning time. Conclusively we find that there is no reason to assume that the Drossel-Schwabl forest fire model develops into a critical state. This is in line with the conclusion by Grassberger [47], who, however, still finds some signs that the forest fire model will finally show some characteristics of standard percolation.

ACKNOWLEDGMENTS

The authors wish to thank Andy Thomas for his technical support. A great deal of the results in this paper were possible only because of his work. This paper partly relied on resources provided by the Imperial College Parallel Computing Centre. We want to thank especially K.M. Sephton for his support. Another part of this work was possible only because of the generous donation made by I-D Media AG, Application Servers & Distributed Applications Architectures, Berlin. We especially thank M. Kaulke and O. Kilian for their support. G.P. wishes to thank P. Grassberger, A. Honecker, I. Peschel, and K. Schenk for very helpful communication, as well as P. Anderson and K. Dahlstedt for their advice. The authors gratefully acknowledge the support of EPSRC.

[1] H. J. Jensen, *Self-Organized Criticality* (Cambridge University Press, New York, 1998).
[2] T. Hwa and M. Kardar, Phys. Rev. Lett. **62**, 1813 (1989).
[3] G. Grinstein, D.-H. Lee, and S. Sachdev, Phys. Rev. Lett. **64**, 1927 (1990).
[4] A. Vespignani and S. Zapperi, Phys. Rev. E **57**, 6345 (1998).
[5] G. Pruessner and H. J. Jensen, Europhys. Lett. **58**, 250 (2002).
[6] B. Drossel and F. Schwabl, Phys. Rev. Lett. **69**, 1629 (1992).

[7] Z. Olami, H. J. S. Feder, and K. Christensen, Phys. Rev. Lett. **68**, 1244 (1992).
[8] S. Lise and M. Paczuski, Phys. Rev. E **63**, 036111 (2001).
[9] S. Lise and M. Paczuski, Phys. Rev. E **64**, 046111 (2001).
[10] C. J. Boulter and G. Miller, Phys. Rev. E **68**, 056108 (2003).
[11] J. J. Binney, N. J. Dowrick, A. J. Fisher, and M. E. J. Newman, *The Theory of Critical Phenomena* (Clarendon, Oxford, 1998).
[12] H. E. Stanley, *Introduction to Phase Transitions and Critical Phenomena* (Oxford University Press, New York, 1971).
[13] T. E. Harris, *The Theory of Branching Processes* (Springer-Verlag, Berlin, 1963).
[14] D. Stauffer and A. Aharony, *Introduction to Percolation Theory* (Taylor & Francis, London, 1994).
[15] S. Clar, B. Drossel, and F. Schwabl, Phys. Rev. E **50**, 1009 (1994).
[16] K. Christensen, H. Flyvbjerg, and Z. Olami, Phys. Rev. Lett. **71**, 2737 (1993).
[17] P. Grassberger, J. Phys. A **26**, 2081 (1993).
[18] G. Pruessner and H. J. Jensen, Phys. Rev. E **65**, 056707 (2002).
[19] P. Bak, K. Chen, and C. Tang, Phys. Lett. A **147**, 297 (1990).
[20] A. Honecker and I. Peschel, Physica A **239**, 509 (1997).
[21] C. L. Henley, Phys. Rev. Lett. **71**, 2741 (1993).
[22] K. Schenk, B. Drossel, S. Clar, and F. Schwabl, Eur. Phys. J. B **15**, 177 (2000).
[23] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms* (MIT Press, Cambridge, MA, 1990), pp. 194 and 912.
[24] S. Clar, B. Drossel, and F. Schwabl, J. Phys. C **8**, 6803 (1996).
[25] R. Pastor-Satorras and A. Vespignani, Phys. Rev. E **61**, 4854 (2000).
[26] F. J. Wegner, Phys. Rev. B **5**, 4529 (1972).
[27] K. Christensen, H. C. Fogedby, and H. J. Jensen, J. Stat. Phys. **63**, 653 (1991).
[28] A. Honecker, Program for simulating the two-dimensional forest-fire model (1997), URL http://www-public.tu-bs.de:8080/~honecker/software/forest2d.h%tml
[29] M. E. J. Newman and R. M. Ziff, Phys. Rev. Lett. **85**, 4104 (2000).
[30] M. E. J. Newman and R. M. Ziff, Phys. Rev. E **64**, 016706 (2001).
[31] N. R. Moloney and G. Pruessner, Phys. Rev. E **67**, 037701 (2003).
[32] J. Hoshen and R. Kopelman, Phys. Rev. B **14**, 3438 (1976).
[33] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms* (MIT Press, Cambridge, MA, 1990).
[34] K. Dowd and C. Severance, *High Performance Computing*, 2nd ed. (O'Reilly, Sebastopol, CA, 1998).
[35] H. Müller-Krumbhaar and K. Binder, J. Stat. Phys. **8**, 1 (1973).
[36] D. P. Landau and K. Binder, *A Guide to Monte Carlo Simulations in Statistical Physics* (Cambridge University Press, Cambridge, UK, 2000).
[37] T. W. Anderson, *The Statistical Analysis of Time Series* (Wiley, London, 1964).
[38] A.M. Ferrenberg, D.P. Landau, and K. Binder, J. Stat. Phys. **63**, 867 (1991).
[39] B. Efron, *The Jackknife, the Bootstrap and Other Resampling Plans* (SIAM, Philadelphia, PA, 1982).
[40] G. Pruessner, D. Loison, and K.-D. Schotte, Phys. Rev. B **64**, 134414 (2001).
[41] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI*, 2nd ed. (MIT Press, Cambridge, MA, 1999).
[42] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, 2nd ed. (Cambridge University Press, New York, 1992).
[43] S. Aluru, G. M. Prabhu, and J. Gustafson, Parallel Comput. **18**, 839 (1992).
[44] P. D. Coddington, NHSE Review™ (1996), URL: http://www.crpc.rice.edu/NHSEreview/RNG/
[45] K. Entacher, Simulation **72**, 163 (1999).
[46] K. Schenk, B. Drossel, and F. Schwabl, Phys. Rev. E **65**, 026135 (2002).
[47] P. Grassberger, New J. Phys. **4**, 17 (2002).
[48] M. De Menech, A. L. Stella, and C. Tebaldi, Phys. Rev. E **58**, R2677 (1998).
[49] C. Tebaldi, M. De Menech, and A. L. Stella, Phys. Rev. Lett. **83**, 3952 (1999).
[50] P. Pfeuty and G. Toulouse, *Introduction to the Renormalization Group and to Critical Phenomena* (Wiley, Chichester, 1977).
[51] P. Sinha-Ray and H. J. Jensen, Phys. Rev. E **62**, 3215 (2000).
[52] In a finite system, the distributions are not expected to be free of any scale, but to be dominated asymptotically by one scale only, which diverges in the thermodynamic limit.
[53] The extreme case would be $\mathsf{P}(s,T_M;\theta)=\delta(s-f(T_M))g(T_M)$ with a monotonic function $f(T_M)$ representing the conditional average.
[54] Similarly for other forms of path compression, for example bending the pointer of the preceding to the adjacent site in find_root (Fig. 8).
[55] For integers the precision is not a problem, but the maximum representable number easily becomes a problem.
[56] One might be inclined to postpone the sending of the borders to a time when it is really needed. However, after a successful burning, the time t is increased and this enters the histogram (see Sec. II C 3). Ignoring this change for a large number of steps would introduce uncontrollable deviations of the estimator of the histogram from its true value.