Spreading and backbond dimensions of 2D percolation

1992 J. Phys. A: Math. Gen. 25 5475

(http://iopscience.iop.org/0305-4470/25/21/009)

J. Phys. A: Math. Gen. 25 (1992) 5475–5484. Printed in the UK

Spreading and backbone dimensions of 2D percolation

P Grassberger

Physics Department, University of Wuppertal, D-5600 Wuppertal 1, Federal Republic of Germany

Received 9 March 1992, in final form 15 June 1992

Abstract. We present results of high-statistics simulations of the spreading of 2D percolation, and of backbones of 2D percolation clusters. While the algorithm employed for the spreading is more or less standard, for identifying backbones we use a new and very efficient recursive algorithm. We use our new critical exponents to test previous conjectures.

1. Introduction

The 'classical' critical exponents for percolation (those which are analogous to the usual critical exponents in thermodynamic spin models, and in the Potts model in particular) are now well understood in two dimensions. This is not to the least due to the conformal invariance which field theories in two dimensions obey.

For percolation there are, however, also a number of other critical exponents which seem to have no analogue in thermodynamics, and which cannot yet be computed exactly, even in two dimensions.

One such exponent is the fractal dimension of the 'backbone' [1]. Consider bond percolation on a square lattice, and assume that all sites on the x-axis and on the opposite edge $x = L$ are wetted. The backbone is then defined as the set of all sites in the interior of the lattice which are connected by conducting bonds to both edges, by two paths which have no edge in common. Except for 'Wheatstone bridges', it consists exactly of those sites through which current would flow if the two edges $x = 0$ and $x = L$ were subject to a potential difference. The average number of sites in the backbone scales at $p = p_c$ as $N_B \sim L^{D_B}$, where D_B is its fractal dimension.

Another important exponent is the 'spreading dimension' [2] $\hat{d}$ defined via the average mass M_t which is connected to some randomly chosen centre by a shortest path of length $< t$. It is defined as $\langle M_t \rangle \sim t^{\hat{d}}$, where the averaging is done over all clusters which contain at least one site whose shortest path from the centre has length $\geqslant t$. It can be interpreted as the size of a region infected within t time steps by an epidemic spreading just marginally from a single infected site, and conditioned on those epidemics which have not yet died out. Related to the spreading dimension are a number of other exponents, as e.g. that giving the growth of the average radius of the cluster of infected sites, or the fractal dimension d_{min} of shortest paths between two randomly chosen points on the incipient infinite cluster [3, 4].

Finally, there are exponents describing the spectral density of the Laplace operator on the fractal, and related exponents describing electric conductivity at and near the critical point [6].

0305-4470/92/215475+10$07.50

It is an important open question whether these sets of exponents are related. For instance, there is a conjecture due to Herrmann and Stanley [5] which says that

$$d_{\min} = 2 - D_B + 1/\nu \tag{1}$$

and which was up to now compatible with the best available estimates. Also, it is not known whether they are rational numbers like thermal exponents in two dimensions.

It is the aim of the present paper to clarify this situation by providing Monte Carlo estimates of these exponents which are substantially more precise than previous ones. While we use an essentially standard Leath-type algorithm for simulating spreading [4], we use a novel and highly non-trivial algorithm for identifying backbones.

2. Spreading

As already stated, we used for spreading essentially the algorithm described in [4]. This consists of starting the 'epidemic' not from a single site, but from one boundary of a rectangular lattice with sidewise periodic boundary conditions, i.e. with the topology of a cylinder. At any 'generation' t, newly wetted sites must be neighbours of sites which had been wetted just in the last generation. We call these latter sites 'growth sites'. In order to enhance the efficiency of the program, we store their coordinates in an array which is updated at every generation.

The observables measured are the number of growth sites N_t and their average distance x_t from the seeding edge. The distance should satisfy the scaling law

$$x_t \sim t^{\nu/\nu_t} \tag{2}$$

where ν_t is related to the thermal exponents, the spreading dimension, and to the fractal dimension d_f by [4, 7]

$$\nu_t = \frac{\nu d_f}{\hat{d}} = \frac{d\nu - \beta}{\hat{d}}. \tag{3}$$

It is related to the shortest path dimension $d_{\min}$ [5] as $d_{\min} = \nu_t/\nu$. The number of growth sites satisfies

$$N_t \sim t^{-z} \tag{4}$$

with

$$z = 1 - \frac{\nu - \beta}{\nu_t}. \tag{5}$$

Compared to [4], the statistics in the present paper are larger by roughly two orders of magnitude. Altogether 15 264 lattices of width 16 384 were simulated for 5 000 generations each. This corresponded to about 1.5×10^{11} wetted sites. This high statistics was possible by using new hardware (mostly a DEC station 2100) and several small improvements in the algorithm which together gave nearly one order of magnitude increase in speed. These improvements involved use of multi-spin coding (1 bit per site), writing both coordinates into one word, and using

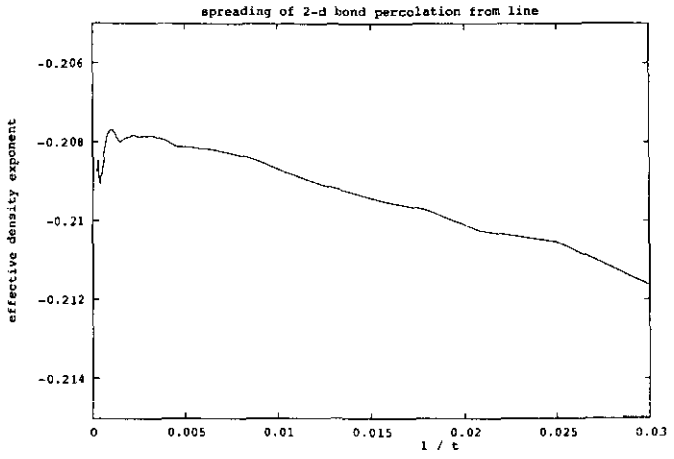

Figure 1. Effective exponent $z_{\text{eff}}(t)$ versus $1/t$ for 2D critical bond percolation.

one single bit of a random word for determining whether a bond was open or not. The latter obviously depended on the fact that we simulated bond percolation where p_c is known to be exactly 1/2. For random number generators (RNG), we used two shift-register generators, namely $i_n = i_{n-137}$ *XOR* i_{n-250} [8] and $i_n = i_{n-157}$ *XOR* i_{n-314} *XOR* i_{n-471} *XOR* i_{n-9689} [9]. Both were implemented with 32-bit integers which reduced the number of RNG calls drastically. The number of wetted sites per second was ca 70.000 on the DEC station (which runs at about 12 MIPS).

When fitting the above scaling laws, we have to take into account two possible sources of error. On the one hand, (2) and (4) give only leading terms, and should in general be supplemented with terms involving smaller exponents ('confluent singularities'). On the other hand, neither the origin of time nor the origin of space are *a priori* fixed to better than roughly one lattice constant and one generation, respectively. Equations (2) and (4) should thus in general be replaced by

$$x_t = x_0 + A(t - t_0)^{\nu/\nu_t}\left(1 + \frac{a_1}{t^{\delta_1}} + \cdots\right) \tag{6}$$

and

$$N_t = B(t - t_0)^{-z}\left(1 + \frac{b_1}{t^{\delta_1}} + \cdots\right). \tag{7}$$

We found that indeed no good fit was obtainable with the uncorrected equations (2) and (4). But all observed corrections to scaling could be absorbed into two constants $x_0 = 0.2$ and $t_0 = 0.62$, and no further confluent singularities were needed. The resulting effective exponents $z_{\text{eff}}(t)$ and $(\nu/\nu_t)_{\text{eff}}(t)$, defined as obtained from least-squares fits over intervals $[t/8, t]$, are plotted in figures 1 and 2. The extrapolations to $t = \infty$ give $z = 0.208 \pm 0.001$ and $\nu/\nu_t = 0.8843 \pm 0.0003$. Using the exact values $\nu = 4/3$, $\beta = 5/36$, we obtain from the latter

$$\nu_t = 1.5078 \pm 0.0005 \qquad d_{\min} = 1.1307 \pm 0.0004 \qquad \hat{d} = 1.6765 \pm 0.0006 \tag{8}$$

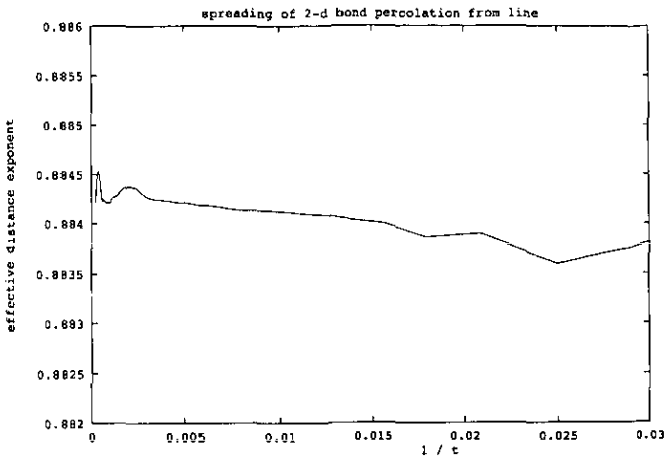

Figure 2. Similar to figure 1, but effective exponent $(\nu/\nu_t)_{\text{eff}}(t)$.

Compared to this, the measured value of z is much less precise, and thus contributes little to the determination of the critical exponents. It provides, however, a very welcome consistency test. Using (5) and (8), we would have predicted $z = 0.2078 \pm 0.0003$ *in obvious agreement with the direct measurement.* This verifies that we have indeed been working at the critical point.

Equation (8) is fully compatible with the most precise previous estimates of ν_t [4, 5], but is one order of magnitude more precise.

3. Backbones

In contrast to the simulations of entire percolation clusters, up to now there did not exist any very efficient algorithm for identifying their backbones. The algorithm used in [1], e.g. seems to have been very slow. Much faster is an algorithm due to Tarjan [10]. It is a recursive depth-first algorithm, and has a time complexity $O(N)$ where N is the number of sites in the cluster. It is not very easy to implement, and it requires a considerable amount of storage since it needs two counters for each occcupied site, in addition to the variables indicating the occupancy of each bond resp. site. Finally, there exists an algorithm due to Roux and Hansen [11]. It also seems not easy to implement. It should be slower than Tarjan's (as one needs several passes through the lattice), but no detailed comparison is available.

The present results are based on a new algorithm which is about twice as fast as Tarjan's, and needs about half as much storage. It is also a recursive depth-first algorithm, but of rather different structure. In contrast to Tarjan's algorithm, it works only for planar graphs, hence it could not be applied to 3D percolation or to 2D percolation on lattices with non-trivial (e.g. cylindrical) topology.

Let us consider site percolation on a square lattice where all sites on the x-axis and on the opposite edge $y = L$ are wetted, while all sites on the side edges $x = 0$ and $x = L$ are non-wetted. The backbone consists then of non-self-intersecting paths connecting the edges $y = 0$ and $y = L$, and containing only occupied sites.

In order to understand our algorithm, let us first discuss a simpler recursive algorithm which just tests whether there is a percolating cluster by giving the left-most connecting path and the tangling parts of the cluster left of it (see figure 3).

This will be part of the final algorithm. It involves four subroutines 'east', 'west', 'south' and 'north', each corresponding to one step in the corresponding direction. By recursive calls, the algorithm generates a branched path which starts at $x = y = 0$ and essentially (though not exactly) follows the left part of the hull of the cluster containing the x axis. A 'C' program for this would look as follows.

```
void north(int x, int y), ... ;
int s[L][L];
int x,y;
float p;
main()
{
    p = .592746;          /*   from ref.[11]  */
    do (x=0;x<L;x++)  north(x,0);
    printf("cluster does not percolate !\n")
}
north(x,y)
{
    if (y<L)
    {
        if (s[x][y]==0)
        {
            s[x][y]=1;
            if (rand()<p)
            {
                west(x-1,y);
                north(x,y+1);
                east(x+1,y);
            }
        }
    }
    else
    {
        printf("cluster percolates !\n");
        exit(0);
    }
}
east(x,y)
{
    if (x<L)
    {
        if (s[x][y]==0)
        {
            s[x][y]=1;
            if (rand()<p)
            {
                north(x,y+1);
                east(x+1,y);
                south(x,y-1);
```

```
            }
        }
    }
}
south(x,y)
{
    if (y>0) { ... }
}
west(x,y)
{
    if (x>0) { ... }
}
```

Notice that here the four subroutines could easily have been combined into a single subroutine, but keeping four distinct subroutines makes the subsequent modifications easier.

The first modification consists of labelling each site not only by `s[x][y]=1` or `s[x][y]=0` for tested/non-tested, but by `s=0` (non-tested), `s=q` (occupied) and `s=INT_MAX` (empty). Here q is any positive integer less than `INT_MAX`. In a second modification, we count the number of sites in the backbone by incrementing a counter 'm' each time when entering a subroutine, and decrementing it when leaving. If the routine stops since it reached the far side $y = L$, then all steps in the backbone correspond to subroutine calls which are not yet exited, and the value of m is just the size of the backbone. After these modifications, subroutine 'west' reads e.g.

```
west(x,y)
{
    if (y>0)
    {
        if (s[x][y]==0)
        {
            if (rand()<p)
            {
                m++;
                s[x][y]=q;
                south(x,y-1);
                west(x-1,y);
                north(x,y+1);
                m--;
            }
            else s[x][y]=INT_MAX;
        }
    }
}
```

The last modifications have to be such as to count also all other connecting paths. For this we first remove the 'exit' in subroutine 'north'. Instead we increase q each time when either the the upper edge is reached, or if q is larger than a tested non-zero `s[x][y]`. Next, we replace the unconditional decrement of m by a decrement

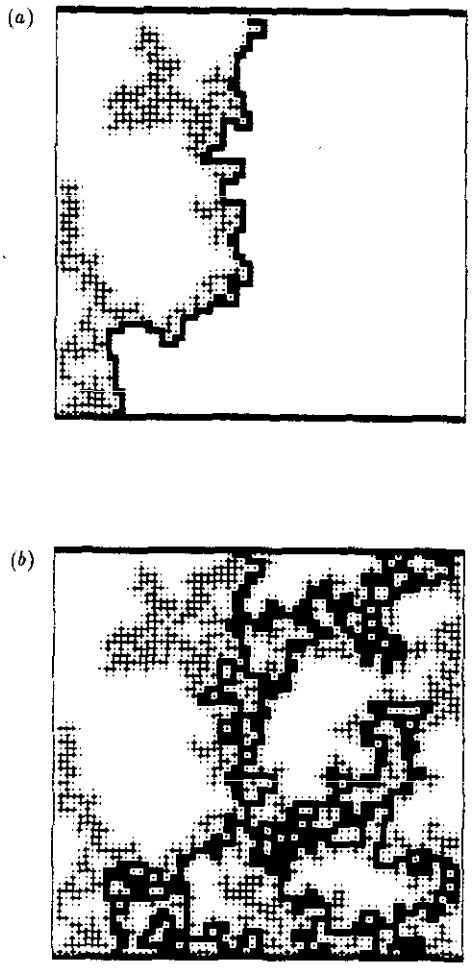

Figure 3. (*a*) Part of the percolating cluster on a 60 x 60 lattice. The construction of the percolating cluster started at the lower left corner. Following a branched random walk, it traced out the left ('western') rim of the cluster. Light points are sites which upon testing have been found to be blocked. Crosses are unblocked sites which belong to tangling ends (and thus do not belong to the backbone). Finally, the line of heavy dots is the leftmost part of the backbone. The figure is taken just before the moment when the walk hits first the upper edge. (*b*) The same cluster after completion.

under the condition that q has not changed since entering the subroutine. This is so since a change of q indicates that either the upper edge had been reached after the subroutine had been entered (in which case the site belongs to the backbone), or that a site belonging to the backbone had been hit (in which case the present site also belongs to the backbone).

In this way we would, however, after first hitting the upper edge, trace out not the most 'western' but the most 'eastern' path back to the lower edge, and we would have trouble to correctly find the inner parts of the backbone. To avoid this, we have to remember whether we are moving upward or downward. In the former case, we take

the left-most branches first, in the latter case we first try the right-most branches. A cheap way to remember the direction of walking consists of encoding it in the evenness/oddness of q. Thus we start with even q (say $q = 2$), increase q to the next higher odd number when hitting the upper edge, increase it to the next higher even number when hitting the edge $x = 0$, and increase it by 2 units when hitting any other backbone site. This brings e.g. subroutine 'west' into the form

```
int m,q;
...
west(x,y)
{
    if (y>0)
    {
        if (s[x][y]==0)
        {
            if (rand()<p)
            {
                m++;
                s[x][y]=q;
                if (q&1==0)
                {
                    south(x,y-1);
                    west(x-1,y);
                    north(x,y+1);
                }
                else
                {
                    north(x,y+1);
                    west(x-1,y);
                    south(x,y-1);
                }
                if (q==s[x][y]) m--;
            }
            else s[x][y]=INT_MAX;
        }
        else if (q>s[x][y]) q+=2;
    }
}
```

Finally, there are a number of minor details (like e.g. emptying the stack after each run) which we do not want to discuss here. The algorithm was first implemented on a home computer where the interactive graphics was essential to get it bug-free. After optimization, it ran at about 43.000 wetted sites per second on the DEC station 2100.

We analysed lattices with $L = 5, 7, 10, 14, 20, 28, \ldots 1280$. For each L, the number of realizations was between $> 10^6$ (for the lattices with $L \leqslant 320$) and about 2×10^5 (for $L = 1280$). The percolation probability P was somewhat larger than 1/2 for all lattice sizes, and seemed to converge rather slowly to 1/2 for $L \to \infty$. Convergence to $P = 1/2$ is indeed expected from universality with bond percolation. For p_c we used the value 0.592 745 of [12].

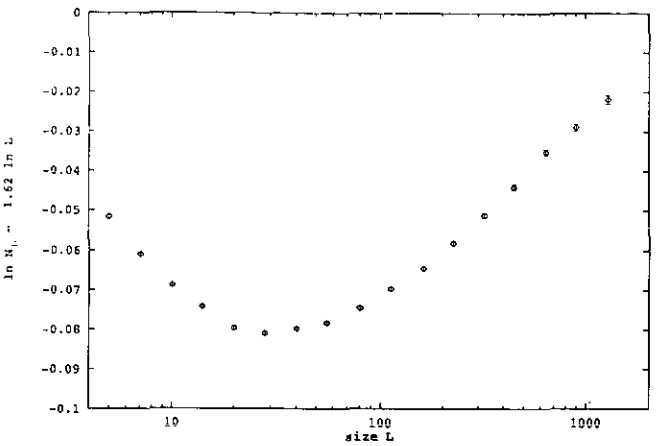

Figure 4. Log–log plot of $\ln(N_B/L^{1.62})$ versus $\ln L$.

The effective dimension of the backbone, defined as average local slope in a log–log plot, was clearly rising with L. For small lattices ($L \sim 30$), we found $D_{B,\text{eff}} \approx 1.62$ as in [1, 13, 14]. But, as seen from figure 4, the extrapolation to $L = \infty$ gives a definitely larger value. A precise extrapolation was made difficult by the large corrections to scaling, and by the slow convergence of P. Fitting an ansatz with a confluent singularity as in (6), we obtain $\delta_1 \approx 0.6$ and

$$D_B = 1.647 \pm 0.004. \tag{9}$$

This is a factor of 5 more precise than the value $D_B = 1.62 \pm 0.02$ of [1, 13], and slightly outside their error bars. The values 1.61 ± 0.01 of [14] and 1.71 ± 0.01 of [15], though formally more precise, seem definitely ruled out.

4. Conclusions

Our simulations provide estimates of two 'non-classic' critical exponents for 2D percolation which are by far the most precise to date.

These estimates (8) and (9) together rule out the Herrmann–Stanley conjecture (1) by about 6 standard deviations. This is one of our main conclusions.

Another motivation was to look for rational expressions of the exponents, as this could give some hope of obtaining them from conformal invariance. Of course it is easy to represent any real number by a rational one, as long as the error bars are not zero. With errors as in (8), one can give already many fractions with denominator and denumerator both less than 100. But we want somewhat more. For instance, a very good approximation to the spreading dimension would be $\hat{d} = 57/34$. Inserting this into (3) would however give $\nu/\nu_t = 1368/1547$, which does not look nice at all. What we want are rational expressions with small denominators for all 3 exponents $\nu_t, \hat{d}, d_{\min}$. This is much harder to find. The only promising candidate we found was

$$\nu_t = 46/39 \qquad d_{\min} = 26/23 \qquad \hat{d} = 161/96. \tag{10}$$

We conjecture that this might indeed be exact. For the backbone dimension, the error is too large for selecting a uniquely looking rational expression.

Acknowledgments

This work was supported by the Deutsche Forschungsgemeinschaft, SFB 238. It is a pleasure to thank Robert Ziff for very useful correspondence.

References

[1] Herrmann H J and Stanley H E 1984 *Phys. Rev. Lett.* **53** 1121
[2] Vannimenus J, Nadal J P and Martin H 1984 *J. Phys. A: Math. Gen.* **17** L351
[3] Grassberger P 1983 *Math. Biosci.* **62** 157
[4] Grassberger P 1985 *J. Phys. A: Math. Gen.* **18** L215
[5] Herrmann H J and Stanley H E 1988 *J. Phys. A: Math. Gen.* **21** L829
[6] Sahimi M and Arbabi S 1990 *J. Stat. Phys.* **62** 453
[7] Grassberger P 1986 *J. Phys. A: Math. Gen.* **19** 1681
[8] Kirkpatrick S and Stoll E P 1981 *J. Comput. Phys.* **40** 517
[9] Ziff R M and Stell G 1988 Critical behavior in three-dimensional percolation: is the percolation threshold a Lifshitz point? *Preprint* University of Michigan, unpublished
[10] Tarjan R 1972 *SIAM J. Comput.* **1** 146
[11] Roux S and Hansen A 1987 *J. Phys. A: Math. Gen.* **20** L1281
[12] Ziff R M and Sapoval B 1986 *J. Phys. A: Math. Gen.* **19** L1169
[13] Nagatani T 1986 *J. Phys. A: Math. Gen.* **19** L1165
[14] Laidlaw D, MacKay G and Jan N 1987 *J. Stat. Phys.* **46** 507
[15] Woo K Y and Lee S B 1991 *J. Phys. A: Math. Gen.* **44** 999