Journal of Statistical Physics, Vol. 66, Nos. 1/2, 1992

Counting Lattice Animals: A Parallel Attack

Stephan Mertens[1] **and Markus E. Lautenbacher**[2]

Received July 15, 1991

A parallel algorithm for the enumeration of isolated connected clusters on a regular lattice is presented. The algorithm has been implemented on 17 RISC-based workstations to calculate the perimeter polynomials for the plane triangular lattice up to clustersize $s = 21$. New data for perimeter polynomials D_s up to D_{21}, total number of clusters g_s up to g_{22}, and coefficients b_r in the low-density series expansion of the mean cluster size up to b_{21} are given.

KEY WORDS: Cluster enumeration; lattice statistics; perimeter polynomials; parallel algorithms.

1. INTRODUCTION

Power series expansions in lattice statistics require the enumeration of finite connected clusters ("lattice animals" or "polyminoes"). The classical site percolation problem[1] considered here is a standard example. If p denotes the probability that a lattice site is occupied, the mean number (per lattice site) of connected clusters of occupied sites with size s is given by

$$n_s(p) = \sum_t g_{st}\, p^s (1-p)^t =: p^s D_s(q) \tag{1}$$

with $q = 1 - p$. In the above equation g_{st} denotes the number of possible clusters with size s and perimeter t and $D_s(q)$ is usually termed the perimeter polynomial. The perimeter polynomials comprise a considerable amount of information about the percolation problem. $D_s(1) =: g_s$ gives the total number of s-clusters per lattice site, for example, and the low-density series expansion of the mean cluster size $S(p) = p^{-1} \sum_s s^2 n_s$ can easily be calculated from the coefficients g_{st}.

[1] Universität Göttingen, Institut für Theoretische Physik, D-3400 Göttingen, Germany.
[2] Technische Universität München, Physik Department T30, D-8046 Garching, Germany.

Quantities like $S(p)$ show nonanalytic behavior at the percolation threshold p_c. To learn something about the location and the nature of such singularities from series expansions, one has to know as many coefficients as possible.[2] To obtain series of reasonable length, the calculation of the g_{st}'s, i.e., the enumeration of lattice animals, has to be put on a computer. Every enumeration algorithm, however, has to cope with the exponential growth of the number of lattice animals with cluster size s: The time required for a complete enumeration of lattice animals up to a size s grows roughly like $\propto \lambda^s$, where λ is somewhat below the coordination number of the lattice. This exponential growth limits the easily reachable size very soon and calls for highly efficient algorithms.

Based on the "shadow" method developed by Sykes,[3] Martin[4] described an algorithm which considerably reduces the maximum cluster size to be enumerated. The method proposed by Sykes, however, applies only to bipartite lattices and in addition requires large amounts of fast computer memory.

On the other hand, the brute-force enumeration of lattice animals by direct counting can deal with any lattice type and its memory requirements can be neglected. However, whether this approach can give useful results despite exponential growth in computer time crucially depends on its effective implementation on a computer.

It is the purpose of this paper to show that calculating the perimeter polynomials by brute-force enumeration can indeed be very efficient, since the counting algorithm can be perfectly parallelized. With its low storage requirements and its parallel nature the algorithm is well suited for medium-sized workstations which are becoming increasingly available at research institutes. E.g., for this study we used 17 RISC-based workstations in parallel to obtain three new perimeter polynomials for the (nonbipartite) plane triangular lattice.

The outline of the paper is as follows: In the second section we give a brief description of how to parallelize the counting algorithm proposed by Mertens.[5] The third section contains some remarks about the performance of the algorithm and a brief discussion of some aspects of the implementation. The fourth section comprises the results obtained for the plane triangular lattice in terms of three new perimeter polynomials.

2. THE PARALLEL ALGORITHM

Based on ideas of Martin[6] and Redelmeier,[7] Mertens[5] presented an efficient algorithm for the enumeration of lattice animals. It recursively generates all clusters up to a given size $s_{\max}$, i.e., given any s-cluster, the algorithm builds all $(s+1)$-clusters by adding one new cluster site. If all possible $(s+1)$-clusters with this particular s-cluster "parent" are generated, a new s-cluster has to be found which again serves as parent for a new generation of $(s+1)$-clusters. This leads to a depth-first traversal of a "family tree" of all clusters. Each child cluster in the tree consists of its parent plus one new cluster site. This new cluster site has to be chosen so that no older brother or ancestor's older brother in the tree contains it. This is to ensure that every node in the tree is different from every other, i.e., every cluster is generated exactly once. The information about which sites are available for younger brothers and their descendants has to be passed from every node to its offspring and its younger brothers. The way this is done is crucial for the performance of the algorithm.

To explain why it can be parallelized, we give here only a simplified description of the algorithm, not going into details, like the important concept of "blocked" sites and efficient data structures (see ref. 5). The sites which are available for a node's offspring and younger brothers are coded in the "untried set." The following routine works, given such an untried set, the size s of the cluster to be generated, and the perimeter t of the parent $(s-1)$-cluster. The steps 1,..., 4 are then repeated until the untried set is exhausted. Each iteration generates a child of the parent and each recursion all offspring of that child.

```
1. Remove an element from the untried set.
2. Determine "free" neighbors of this point; $nn :=$ number of these
   new neighbor sites.
3. Count new cluster: Increase $g_{s,t+nn-1}$ by one.
4. If $s < s_{\max}$:
   (a) Add "free" neighbors to the untried set and label corre-
       sponding lattice site "nonfree."
   (b) Call this routine recursively with the current untried set,
       $t := t + nn - 1$ and $s := s + 1$.
   (c) Remove new neighbors from the untried set and relabel
       corresponding lattice sites "free."
```

The key to parallelism lies in step 4c: Here the algorithm drops all information about the recursively generated (and counted) offspring of a node. It has no effect on the rest of the enumeration whether step 4 is actually performed or not! This means that the enumeration of the offspring of some s_0-cluster can be done independently from the enumeration of the offspring of all other s_0-clusters—the "cousins" decouple, as it were. Each subtask of enumerating down from the s_0-cluster to full depth of the tree $(s = s_{\max})$ can be executed on a separate computer. Because of this complete decoupling of subtasks, the problem of lattice animal enumeration considered here is one of the rare cases where the simultaneous use of N processing units reduces the required time by a factor N, since no interprocess communication is needed at all. Therefore we do not even need a truly parallel computer architecture, but instead can distribute the independent subtasks on several ordinary standalone workstations.

In practice we proceed as follows; We choose an s_0 with a moderate number of clusters, say $s_0 = 8$ with 16,689 clusters on the plane triangular lattice (see Table III). If we have N computers at our disposal, we group these clusters in N groups of $n_0 = [16{,}689/N]$ clusters each and associate one group with each computer. For convenience we choose these groups according to the order in which the s_0-clusters are generated in building up the family tree: The first n_0 s_0-clusters belong to the first group, the second n_0 s_0-clusters to the second group, and so on. Every computer explores the family tree up to size s_0, numbering the s_0-clusters according to their order of appearance.[3] If an s_0-cluster belongs to the group with which the computer is associated, the offspring of this cluster is explored to full depth, i.e., $s = s_{\max}$, and the corresponding values of the g_{st}'s are written into a file. After all jobs have been completed, one simply adds corresponding coefficients g_{st} calculated on different computers to get the final result.

The only synchronization required by this parallel organization is to tell each job which part of the s_0-generation has to be explored to full depth. But once the job has been started, no further communication with other jobs is necessary. This facilitates the use of completely independent computers. For example, the results presented here have been found in this way using workstations from different local clusters in Göttingen and Munich.

3. PROGRAMMING ASPECTS AND PERFORMANCE

We think it is worthwhile to spend a few words on how the actual implementation of the algorithm was done. Since CPU time obviously is the bottleneck in lattice animal enumeration, improving the efficiency of the code as much as possible is a major and mandatory task when putting the algorithm on computer. The enumeration algorithm itself together with a simple (nonparallel) FORTRAN implementation has already been described by Mertens elsewhere.[(5)] The program we used here was written in the C programming language, since the recursive formulation of the

[3] Since for $s_0 = 8$ we have $g_{s_0} = 16{,}689$, this repeated initial tree exploration causes negligible overhead, as can be seen form the average cluster counting rate of $\tau = 2.7 \times 10^5$ clusters/sec in Table I.

Table I. Generating Rates in Units of clusters/second on Various UNIX Workstations[a]

Platform	SS 1	DS 3100	SS 2	HP 720
Counting rate	2.45×10^5	2.88×10^5	5.11×10^5	8.12×10^5

[a] The actual data acquisition was done on machines of type DEC station 3100 and Sun SPARCstation 1 only. SS, DS, and HP stand for Sun SPARCstation, DEC station, and Hewlett-Packard, respectively.

Table II. Perimeter Polynomials D_{19}, D_{20}, and D_{21} for the Plane Triangular Lattice[a]

t	$1 = 19$	$s = 20$	$s = 21$
18	1	0	0
19	198	42	6
20	3273	1449	507
21	31288	16461	8292
22	206904	134598	793309
23	1138098	820623	563106
24	5159312	4207833	3226522
25	20570076	18503823	15633920
26	72477567	71501994	66567108
27	228644072	248501115	251246774
28	653800881	776333631	858205000
29	1687724526	2210940684	2661156060
30	3965625385	5734957758	7545172940
31	8449683798	13584271758	19621212282
32	16312020225	29414551056	46836042306
33	28486601108	58079623302	102711592570
34	44734046784	104448199773	206762868636
35	62841331056	170528427444	381403939182
36	78293701534	251442517179	643210904138
37	85253400810	332825218725	988314164230
38	79474311348	391705437144	1376151103032
39	61347762286	403694823051	1724915125136
40	36878757573	356544784128	1925965285922
41	15297106452	260634432204	1885718879538
42	3262576960	148150385331	1583043419972
43	0	58016840826	1099195790960
44	0	11669119236	592621285797
45	0	0	219802218854
46	0	0	41828064480

[a] Perimeter polynomials for smaller values of s can be found in refs. 5, 8, and 9.

algorithm does not fit nicely into a non-recursive language like FORTRAN, and the UNIX operating system clearly favors C. After the program had been written, we used the various profiling tools available to locate the "hotspots" in our code where most of the execution time was spent. These "hotspots" were optimized by standard techniques, such as explicitly unrolling loops and if-then-else structures. Then in a refined profiling analysis we searched for the handfull of variables accessed most, in our case certain array indices used to address perimeter polynomial coefficients g_{st}. We declared some of these variables to be of storage class "register." Thus, there variables would be kept in one of the CPU's register during the whole calculation instead of being swapped from and to the memory everytime they were accessed. Finding out the optimal choice and number of "register" variables by trial-and-error after explicit code un-

Table III. Total Number g_s of Clusters Grouped by Sites on the Plane Triangular Lattice[a]

s	g_s
1	1
2	3
3	11
4	44
5	186
6	814
7	3652
8	16689
9	77359
10	362671
11	1716033
12	8182213
13	39267086
14	189492795
15	918837374
16	4474080844
17	21866153748
18	107217298977
19	527266673134
20	2599804551168
21	12849503756579
22	63646233127758

[a] g_{21} and g_{22} are new. g_{20} has been derived recently by Sykes and Flesia[11] using older enumeration data and combinatorial arguments. Their value is confirmed by our direct enumeration.

rolling, we were able to achieve an overall performance increase of 40% in terms of counted clusters per second relative to the nonoptimized version of the program. Having in mind the overall computer resource consumption of the present lattice animal study (see discussion below), we think the time that went into program optimization was well spent after all.

To give some explicit ratings on program performance, we have compiled in Table I the cluster generating rates per second as measured on a number of workstations to which we had access. However, actual calculations that went into the present lattice animals data were only done on two of them, the Sun SPARCstation 1 and the DEC station 3100, both of which are machines based on modern RISC technology. With the total number of clusters counted up to a maximum cluster size of $S_{\max} = 21$ being $g_{\text{tot}} = 16111290471381$ and an average generating rate of $\rho \approx 2.7 \times 10^5$ clusters/sec (see Tables III and I, respectively) one readily derives a CPU

Table IV. Coefficients for the Expansion of $s(p) = \sum_r b_r p^r$ [a]

r	b_r
1	6
2	18
3	48
4	126
5	300
6	750
7	1686
8	4074
9	8868
10	20892
11	44634
12	103392
13	216348
14	499908
15	1017780
16	2383596
17	4648470
18	11271102
19	20763036
20	52671018
21	91377918

[a] b_{20} and b_{21} are new to us; b_{19} has again been calculated recently by Sykes and Flesia[11] using combinatorics and is confirmed by our direct enumeration.

time consumption of $\tau = g_{\text{tot}}/\rho \approx 6.44 \times 10^7$ sec ≈ 746 days ≈ 2.04 years for the whole study. By splitting the complete task of cluster counting into smaller independent ones as described in Section 2 and running the subtasks on about 17 different workstations simultaneously, we were able to bring down the time necessary to complete the study by more than one order of magnitude, to a still large but bearable $\tau \approx 44$ says. In reality, the complete enumeration was finished in less than 2 months.

4. RESULTS

Table II shows the perimeter polynomials D_{19}, D_{20}, and D_{21} for the plane triangular lattice which are new to us. Perimeter polynomials for smaller values of s can be found in refs. 5, 8, and 9. We have used our data to calculate the total number g_s of s-clusters (Table III) and the coefficients b_r of the low-density series expansion of the mean cluster size $S(p) = \sum_r b_r p^r$ (Table IV). The knowledge of the perimeter polynomials up to size $s_{\max}$ allows the calculation of g_s up to $s = s_{\max} + 1$ and the series expansion of S up to order $s_{\max}$. For lattices for which the low-density expansion of the mean number of clusters

$$K(p) = \sum_r k_r p^r \tag{2}$$

is available through $k_{s_{\max}+2}$, the perimeter polynomials of size $s_{\max}$ determine both $b_{s_{\max}+1}$ and $g_{s_{\max}+2}$.[10] This fact has been exploited by Sykes and Flesia[11] to obtain g_{20} and b_{19}. Their values have been confirmed by our direct enumeration.

It is interesting to note that the coefficients b_r in the expansion of $S(p)$ for the triangular lattice keep growing monotonically with r. This should be compared to the corresponding coefficients b_r of the simple square lattice, which oscillate for larger values of r.[5]

5. CONCLUSIONS

In this study we have shown that the enumeration of lattice animals can be organized in a parallel algorithm which reduces the necessary time by a factor N, where N is the number of available computers. This is the maximum improvement that can be expected from a parallel algorithm using N processors, since in the form presented here the algorithm requires no interprocess communication at all. Facing the exponentially growing complexity of the enumeration problem, this might be regarded as only a modest advance. Nevertheless the proposed algorithm already yields the maximum speedup possible when trying to attack the problem with multiple computers instead of one computer. Along these lines one simply cannot do better. Further improvement calls for different methods.

The relevance of larger and larger exact enumerations may well be questioned, of course. However, besides the pure academic motivation of knowing still a few perimeter polynomials more, there is hope that if reaching large enough values of cluster size s exact enumerations like this one can give valuable guidance in the search for some analytic asymptotic theory of lattice animals which, once found, surely will supplement exact enumerations with a deeper understanding of lattice animal growth. Also, numerical estimates for scaling corrections become more reliable with the knowledge of additional perimeter polynomials.

It should be mentioned, however, that our parallel algorithm in the present formulation is restricted to the calculation of perimeter polynomials. A more general method for the effective generation of series expansions, the "no-free-end" method, can be found in ref. 12.

In addition to the enumeration results presented, this paper can be regarded as another example of how to attack a research problem in computational physics using only medium-sized workstations instead of the expensive CPU time of sometimes rather user-unfriendly and little flexible mainframes. With prices falling, the number of workstations available is expected to increase rapidly in the future. However, a considerable percentage of them probably will be used only interactively for tasks like text-processing, graphics, or e-mail, thus leaving the powerful CPU almost idle. We have demonstrated here that on such machines a long-running, low-priority background job which usually is not even recognized by the average user is able to pick up enough CPU time to yield research results in a reasonable amount of time if the whole project is coordinated properly by running similar jobs on other idle workstations. We believe that along these lines the computational power of the growing number of workstations may well be used to attack a number of problems which are up to now in the realm of mainframes and supercomputers.

ACKNOWLEDGMENTS

We would like to thank D. Stauffer for his helpful suggestions on the manuscript. All calculations have been performed on workstations of the Institute of Numerical and Applied Mathematics in Göttingen and the Institute for Theoretical Physics of the Physics Department at Technical University Munich. One of us (S.M.) appreciates the support of the local system administrator G. Siebrasse.

REFERENCES

1. D. Stauffer, *Introduction to Percolation Theory* (Taylor & Francis, London, 1985), and references therein.
2. C. Domb and M. S. Green, eds., *Phase Transitions and Critical Phenomena*, Vol. 3 (Academic Press, 1974).
3. M. F. Sykes, *J. Phys. A Math. Gen.* **19**:1007–1025, 1027–1032, 2425–12429, 2431–2437 (1986).
4. J. L. Martin, *J. Stat. Phys.* **58**:749 (1990).
5. S. Mertens, *J. Stat. Phys.* **58**:1095 (1990).
6. J. L. Martin, in *Phase Transitions and Critical Phenomena*, Vol. 3, C. Domb and M. S. Green, eds. (Academic Press, 1974), pp. 97–112; see S. Redner, *J. Stat. Phys.* **29**:309 (1981) for a FORTRAN program.
7. D. H. Redelmeier, *Discr. Math.* **36**:191 (1981).
8. M. F. Sykes and M. Glen, *J. Phys. A Math. Gen.* **9**:87 (1976).
9. A. Margolina, Z. V. Djordjevic, D. Stauffer, and H. E. Stanley, *Phys. Rev. B* **28**:1652 (1983).
10. M. F. Sykes and M. K. Wilkinson, *J. Phys. A* **19**:3415 (1986).
11. M. F. Sykes and S. Flesia, *J. Stat. Phys.* **63**:487 (1991).
12. J. Adler, Y. Meir, A. Aharony, A. B. Harris, and L. Klein, *J. Stat. Phys.* **58**:511 (1990).

Communicated by D. Stauffer