Lower bound on the connective constant for square lattice self-avoiding walks

J. Phys. A: Math. Gen. 26 (1993) 3719–3724. Printed in the UK

Lower bound on the connective constant for square lattice self-avoiding walks

A R Conway† and A J Guttmann‡

† Department of Mathematics, The University of Melbourne, Parkville 3052, Australia
‡ Department of Theoretical Physics, Oxford University, 1 Keble Road, Oxford OX1 3NP, UK

Received 20 November 1992

Abstract. By enumerating irreducible bridges exactly up to 40 steps, and by obtaining a lower bound to the number of bridges with less than 125 steps, a lower bound for the connective constant for square lattice self-avoiding walks of 2.62 is obtained.

1. Introduction

The first study of the lower bound on the connective constant of square lattice self-avoiding walks was by Wakefield, in 1951 [1], and was followed in 1959 by better bounds obtained by Fisher and Sykes [2]. In 1963 Kesten [3] proved a result that allows lower bounds to be established, provided the number of graphs known as *irreducible bridges* is known. This result was used by Beyer and Wells [4] in 1972 to obtain a better lower bound, which was improved by Guttmann [5] in 1983. This gave $\mu > 2.568$, a result that was recently substantially improved by Alm [6], who proved that $\mu > 2.601\,774$. In this paper we have improved on this bound, and show that $\mu > 2.62$, which is less than 0.7% below the best numerical estimate [7] of 2.638 158 53.

An n-step self-avoiding walk (SAW) is a non-cyclic continuous path connecting $n + 1$ adjacent vertices on a lattice. If c_n denotes the number of n-step SAWs per lattice site, then we define the connective constant, usually denoted μ, by $\lim_{n\to\infty} \ln(c_n)/n = \mu$. The existence of this limit was proved [8] in 1954.

For the ease of visualization, we will restrict our subsequent discussion to SAWs on the square lattice, with walk steps in directions parallel to the lattice axes. *Terminally attached walks* (TAWs) are SAWs whose first step is along the positive x-axis, and which subsequently never have x co-ordinate less than that of the end-point of the first step. That is to say, the origin is the *unique* left-most point of the walk. We next define *bridges* as TAWs whose end-point has x coordinate equal to the maximal x coordinate. Then *irreducible bridges* are defined to be bridges which cannot be decomposed into two concatenated bridges. Thus irreducible bridges must have at least three horizontal bonds for each $(x, x + 1)$ segment except the first. See figure 1 for examples. Clearly, each class defined is a subset of the previous class. What is less obvious, though is proved in [5], is that the connective constant for all classes is the same as for unconstrained SAWs.

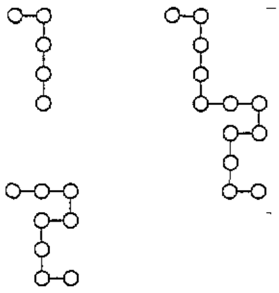

Figure 1. Example of two irreducible bridges (left) and the reducible bridge caused by concatenating them (right).

2. Calculation of lower bound

The method used to establish the new lower bound is the same as that used in [5], and is based upon an observation of Kesten [3]. The generating function for reducible bridges $N(x) = \sum b_n x^n$ is related to the generating function for irreducible bridges $\Lambda(x) = \sum s_n x^n$ by $N(x) = [1 - \Lambda(x)]^{-1}$. As both $N(x)$ and $\Lambda(x)$ have the same radius of convergence, $1/\mu$, it follows [3] that the solution of $\Lambda(x) = 1$ is $x = 1/\mu$. A corollary [3] is that a lower bound for the connective constant μ for SAWs is given by the unique positive root μ_N of the equation

$$f_N(\mu_N) \equiv \sum_{n=1}^{N} s_n \mu_N^{-n} = 1. \tag{2.1}$$

Since $s_n > 0$ for all n, the roots μ_N form an increasing sequence and satisfy $\mu_N \leqslant \mu$. Here s_n is the number of irreducible bridges of n steps.

Note that for $\mu_N > 0$, $f_N(\mu_N)$ is a strictly monotonically decreasing function for $N > 0$, as $s_n > 0$ for $n > 0$. Furthermore, $f_N(\mu_N)$ will drop from ∞ to 0 as μ_N varies from 0 to ∞, and $f_{N+1}(x) > f_N(x)$ for $x > 0$, $N > 0$. This means that $\mu_{N+1} > \mu_N$ for $N > 0$. If there is another function $g_N(x)$ which also decreases monotonically like $f_N(x)$, and which satisfies $f_N(x) \geqslant g_N(x)$, then the positive solution of

$$g_N(\bar{\mu}_N) = 1 \tag{2.2}$$

will have

$$\bar{\mu}_N \leqslant \mu_N \leqslant \mu. \tag{2.3}$$

If $g_N(x)$ is defined by

$$g_N(x) \equiv \sum_{n=1}^{N} \bar{s}_n x^{-n} \tag{2.4}$$

where $0 < \bar{s}_n \leqslant s_n$ for $n > 0$, then this clearly is a suitable function. A useful choice for $\bar{s}_n$ is the number of irreducible bridges of width no more than W (or a lower bound on this number). This clearly satisfies $0 < \bar{s}_n \leqslant s_n$, and has the advantage that this series can, in *principle, be evaluated exactly by using a transfer matrix.*

In practice, evaluating this exactly for large W is a difficult task, as the (square) transfer matrix has side length growing like 3^W. An alternative is to count values of $\bar{s}_n$ on a finite matrix, as was done in [7] to count SAWs on a square lattice to 39 steps. A similar method and algorithm is used here, and is described in section 3.

In summary, to obtain a lower bound $\bar{\mu}_N$ we calculate a lower bound for the number of irreducible bridges of length n and less to get $\bar{s}_1$ to $\bar{s}_N$, and then $\bar{\mu}_N$ is the positive solution to:

$$\sum_{n=1}^{N} \bar{s}_n \bar{\mu}_N^{-n} = 1. \tag{2.5}$$

3. Method of counting

All walks on a rectangular grid W units wide and L units long subject to certain boundary conditions can be counted by using the transfer matrix method described in detail in [7]. A brief summary of the relevant material follows, with the differences for counting irreducible bridges only described.

One starts off with a dividing line on a square grid, with initially all the lattice points to the right. One then moves the dividing line one lattice point at a time to the right, for a total of $(W+1) \times (L+1)$ moves. At each of these moves, a generating function (a polynomial kept to some degree D) for each possible boundary condition is updated. When appropriate boundary conditions are met, the total is accumulated into a resultant generating function. An example of this process is given in figure 2, mid-way through processing the lattice. One permanent boundary condition is that all walks must touch the end of the grid to ensure uniqueness in the horizontal direction.

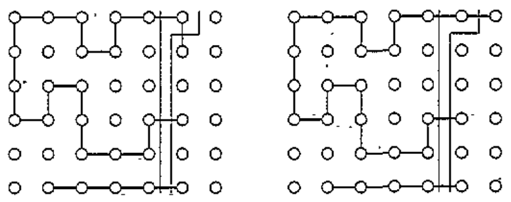

Figure 2. Example of the transfer matrix and finite lattice method of counting bridges. The left boundary is the original boundary. The right boundary shows how a new bond (the one that is different in the two pictures) could be added after adding in an extra site. There are three ways of moving the boundary. Two are shown here: the third is to have no extra bond added. Note that the information to the left of the boundary is never stored, just the boundary itself and the number of bonds to the left.

By keeping the line as close to vertical as possible, the boundary is restricted to crossing a maximum of $W+2$ bonds. The computational complexity in terms of both time and memory of the counting is proportional to a polynomial times 3^W. Thus it is desired to keep W as small as possible.

In [7] two variable generating functions were used, as then use could be made of the symmetry of the horizontal and vertical directions to double the number of terms calculated. For calculating irreducible bridges, this is no longer possible, as there are two distinct directions: parallel to the direction of the 'bridge', and perpendicular to it. For this reason, the generating function stored is only a polynomial in one variable. This reduces the storage requirements significantly, and allows larger values of W to be used than in the SAW case, which partially compensates for the loss of terms from the use of symmetry.

This process is performed for values of W ranging from 0 up to some maximum (10 or 11 in the case of this paper), and with two different boundary conditions: firstly with walks that can only start or finish on the top of the grid, and secondly with the boundary condition that walks can only start of finish on either side of the grid. By subtracting twice the former from the latter, one ends up with the generating function for walks starting at one end and finishing at the other.

We find it more efficient to carry out the above calculations in two steps, one for each boundary condition. The alternative involves a flag tracking the last boundary encountered, and requires up to twice as much storage as the method used, for little time saving. With the transfer matrix method, memory is usually a larger problem than time.

This calculation gives all bridges. We now define a generating function $X^*(u,z)$ where the power of u gives the number of bridges from one side to the other of a lattice of width given by the power of z. This can be converted into the equivalent generating function for irreducible walks $X(u,z)$ efficiently via the relation

$$X(u,z) = X^*(u,z)/[1+uzX^*(u,z)] \tag{3.1}$$

using an expansion in z. This will be accurate up to the same width and length as $X^*(u,z)$, and should be truncated after this.

Then, a new generating function $X(u) = X(u,1)$ will be formed. This will have coefficients which provide a lower bound on the number of irreducible bridges fitting into the grid. Thus the coefficients in $uX(u)$ can be used in (2.4) as the coefficients $\bar{s}_n$. A lower bound for μ can then be obtained. The u multiplying $uX(u)$ is to account for the extra step required at the start of the walk to keep the notation the same as used in [5].

A merely implementational difficulty is the fact that many of the numbers involved are large, and it is important to know them exactly. These numbers can become too large to be efficiently stored or handled by the machine. They also increase the memory requirements. The normal solution of using modular arithmetic, and repeating the calculations for various primes was used. This is easy since no divisions ever need to be done.

The calculation was performed up to widths of 11, and lengths of 80. Eight primes were used in the first stage (obtaining X^*), and 9 were used in obtaining X from X^*.

This process was repeated to widths of 10 and lengths of 124, with ten moduli to gain some extra terms without using too much memory and time.

4. Correction terms

The values obtained for $\bar{s}_n$ by the method above for a maximum grid width of W and a sufficiently large value of L and D (hereafter assumed) will be exact up to but not including bridges of total length $3(W+1)+2+1 = 3W+6$. This will be caused by the bridge *going straight up to $W+2$, across one, down $W+1$, across one and up $W+1$, and by its* mirror image. Thus the first incorrect value will be out by 2.

A lower bound on the correction terms for width $W+1$ will be given by twice the number of bridges formed by a staircase going up to the right, then down, then up again. The factor of two is again due to the fact that they can go to the left or the right.

Each of the $3W+4$ possible sites for adding a number of horizontal bonds will provide a factor of $1+u+u^2+u^3+\cdots = 1/[1-u]$ for this correction generating function, for a total of

$$2u^{3W+6}\frac{1}{[1-u]^{3W+4}}. \tag{4.1}$$

Table 1. Number of irreducible bridges ($n \leqslant 40$) or a lower bound thereof ($n > 40$), plus induced lower bound on the SAW connective constant ($\bar{\mu}_n$).

n	$\bar{s}_n$	$\bar{\mu}_n$
0	0	—
1	1	1.000 00
2	2	2.000 00
3	2	2.269 53
4	2	2.359 30
5	2	2.392 46
6	4	2.417 78
7	10	2.441 60
8	26	2.464 43
9	56	2.482 40
10	118	2.496 31
11	256	2.507 47
12	586	2.516 94
13	1386	2.525 25
14	3262	2.532 51
15	7690	2.538 86
16	18 206	2.544 45
17	43 520	2.549 43
18	104 892	2.553 90
19	254 040	2.557 93
20	614 440	2.561 59
21	1505 906	2.564 92
22	3687 276	2.567 97
23	9061 272	2.570 76
24	22 341 940	2.573 34
25	55 239 616	2.575 72
26	136 930 354	2.577 94
27	340 232 934	2.579 99
28	847 283 502	2.581 91
29	2114 382 206	2.583 70
30	5286 567 268	2.585 39
31	13 241 660 012	2.586 96
32	33 222 895 410	2.588 45
33	83 486 821 126	2.589 85
34	210 106 733 540	2.591 17
35	529 501 940 578	2.592 42
36	1336 175 748 624	2.593 60
37	3375 961 081 570	2.594 73
38	8539 629 649 384	2.595 79
39	21 625 309 327 132	2.596 81
40	54 820 398 745 474	2.597 78
45	5820 318 155 157 312	2.602 00
50	630 290 451 525 202 906	2.605 41
55	69 343 967 856 914 517 780	2.608 22
60	7 719 762 195 096 059 791 214	2.610 58
65	863 967 704 691 674 713 386 386	2.612 59
70	96 310 546 601 365 415 953 964 424	2.614 29
75	10 601 268 097 334 315 639 514 026 656	2.615 74
80	1 146 238 958 432 352 486 626 315 028 232	2.616 94
85	95 122 595 252 413 244 811 302 335 994 414	2.617 78
90	9353 393 218 092 323 606 905 888 547 359 200	2.618 40
95	906 943 093 517 100 736 846 527 879 329 091 430	2.618 87
100	86 955 109 694 998 794 027 177 768 234 060 590 918	2.619 17

Table 1. (continued)

n	$\bar{s}_n$	$\bar{\mu}_n$
105	8261 450 207 357 657 977 290 310 089 675 791 654 798	2.619 51
110	779 111 239 518 194 561 637 275 214 705 111 280 361 530	2.619 71
115	73 029 264 454 070 005 969 830 070 906 741 808 537 647 294	2.619 87
120	6 810 768 067 294 571 953 459 572 425 256 758 676 211 656 394	2.619 99
124	255 630 454 763 564 394 929 879 331 890 351 601 304 148 172 618	2.620 06

This gives the exact correction term to the second incorrect value: $6W + 8$, or in this particular case 74 for ($W = 11$). The next correction term misses bulges in the vertical lines.

These two correction terms make the results exact up to and including $3W + 7$.

Adding the generating function in (4.1) makes a difference of about one part in a billion to the final lower bound, so correction terms have not been pursued further.

5. Results

The first set of the calculations (width 11) took a few days and about 90 megabytes of memory. They were performed on an IBM RS6000/550 with 128 MB of memory.

The second set (width 10, longer series) took about a week of CPU time in a slower IBM RS6000/530 with 192 MB of memory.

The numbers used for $\bar{s}_n$ were the values from the width 11 calculation up to the eighty first term, and values from the width ten run thereafter. Correction terms were included, though they did not make a significant difference. As a result, exact values of $\bar{s}_n$ were used up to the fortieth coefficient, and a lower bound thereafter.

The values of $\bar{\mu}_n$ from (2.2) and (2.4) were calculated via a small Mathematica program. A table of these values is given in table 1.

This gives a new lower bound of 2.62. This is to be compared with the best numerical estimate [7] of 2.638 158 53.

Acknowledgments

We would like to thank Ian G Enting for his contribution in developing the finite lattice method. One of us (ARC) would like to thank the A O Capell, Wyselaskie and Daniel Curdie scholarships. The other (AJG) would like to thank the ARC for financial support. Some of these calculations were performed on an IBM RS6000/550 with 128 Mbytes of memory, kindly loaned by IBM Australia.

References

[1] Wakefield A J 1951 *PhD thesis* Oxford University
[2] Fisher M E and Sykes M F 1959 *Phys. Rev.* **114** 45–58
[3] Kesten H 1963 *J. Math. Phys.* **4** 960–9
[4] Beyer W E and Wells M B 1972 *J. Comb. Theory* A **13** 176
[5] Guttmann A J 1983 *J. Phys. A: Math. Gen.* **16** 2233–8
[6] Alm S E 1992 Private communication
[7] Conway A R, Enting I G and Guttmann A J 1993 *J. Phys. A: Math. Gen.* to be published
[8] Hammersley J 1957 *Proc. Camb. Phil. Soc.* **53** 642