Self-avoiding polygons on the square lattice

1999 J. Phys. A: Math. Gen. 32 4867

(http://iopscience.iop.org/0305-4470/32/26/305)

J. Phys. A: Math. Gen. **32** (1999) 4867–4876. Printed in the UK PII: S0305-4470(99)02204-0

Self-avoiding polygons on the square lattice

Iwan Jensen and Anthony J Guttmann

Department of Mathematics and Statistics, The University of Melbourne, Parkville, Victoria 3052, Australia

Received 25 February 1999, in final form 30 April 1999

Abstract. We have developed an improved algorithm that allows us to enumerate the number of self-avoiding polygons on the square lattice to perimeter length 90. Analysis of the resulting series yields very accurate estimates of the connective constant $\mu = 2.638\,158\,529\,27(1)$ (biased) and the critical exponent $\alpha = 0.500\,0005(10)$ (unbiased). The critical point is indistinguishable from a root of the polynomial $581x^4 + 7x^2 - 13 = 0$. An asymptotic expansion for the coefficients is given for all n. There is strong evidence for the absence of any non-analytic correction-to-scaling exponent.

1. Introduction

A self-avoiding polygon (SAP) can be defined as a walk on a lattice which returns to the origin and has no other self-intersections. The history and significance of this problem is nicely discussed in [14]. Alternatively, we can define a SAP as a connected sub-graph (of a lattice) whose vertices are of degree 0 or 2. Generally SAPs are considered distinct up to a translation, so if there are p_n SAPs of length n there are $2np_n$ walks (the factor of two arising since the walk can go in two directions). The enumeration of SAPs on various lattices is an interesting combinatorial problem in its own right, and is also of considerable importance in the statistical mechanics of lattice models [14].

The basic problem is the calculation of the generating function

$$P(x) = \sum_n p_{2n}x^{2n} \sim A(x) + B(x)(1 - x^2/x_c^2)^{2-\alpha} \tag{1}$$

where the functions A and B are believed to be regular in the vicinity of x_c. We discuss this point further in section 3, as it pertains to the presence or otherwise of a non-analytic correction-to-scaling term. Despite strenuous effort over the past 50 years or so this problem has not been solved on any regular two-dimensional lattice. However, much progress has been made in the study of various restricted classes of polygons and many problems have been solved exactly. These include staircase polygons [26, 25, 5, 2, 20], convex polygons [5, 16, 12, 19], row-convex polygons [2, 20], and almost convex polygons [21]. Also, for the hexagonal lattice the critical point, $x_c^2 = 1/(2 + \sqrt{2})$ as well as the critical exponent $\alpha = \frac{1}{2}$ are known exactly [23, 1], though non-rigorously. Very firm evidence exists from previous numerical work that the exponent α is universal and thus equals $\frac{1}{2}$ for all two-dimensional lattices [11, 9, 15]. Thus the major remaining problem, short of an exact solution, is the calculation of x_c for various lattices. Recently, the authors found a simple mapping between the generating function for SAPs on the hexagonal lattice and the generating function for SAPs on the (3.12^2) lattice [15].

0305-4470/99/264867+10$30.00

Knowledge of the exact value for x_c on the hexagonal lattice resulted in the exact determination of the critical point on the (3.12^2) lattice.

In order to study this and related systems, when an exact solution cannot be found one has to resort to numerical methods. For many problems the method of series expansions is by far the most powerful method of approximation. For other problems Monte Carlo methods are superior. For the analysis of $P(x)$, series analysis is undoubtedly the most appropriate choice. This method consists of calculating the first coefficients in the expansion of the generating function. Given such a series, using the numerical technique known as differential approximants [13], highly accurate estimates can frequently be obtained for the critical point and exponents, as well as the location and critical exponents of possible non-physical singularities.

This paper builds on the work of Enting [7] who enumerated square-lattice polygons to 38 steps using the finite-lattice method. Using the same technique this enumeration was extended by Enting and Guttmann to 46 steps [8] and later to 56 steps [11]. Since then they extended the enumeration to 70 steps in unpublished work. These extensions to the enumeration were largely made possible by improved computer technology. In this work we have improved the algorithm and extended the enumeration to 90 steps while using essentially the same computational resources used to obtain polygons to 70 steps.

The difficulty in the enumeration of most interesting lattice problems is that, computationally, they are of exponential complexity. It would be a great breakthrough if a polynomial time algorithm could be found, while a linear time algorithm is, to all intents and purposes, equivalent to an exact solution. Initial efforts at computer enumeration of square-lattice polygons were based on direct counting. The computational complexity was proportional to λ_1^n, where n is the length of the polygon, and $\lambda_1 = 1/x_c \approx 2.638$. The dramatic improvement achieved [7] by the finite-lattice method can be seen from its complexity, which is proportional to λ_2^n, where $\lambda_2 = 3^{\frac{1}{4}} \approx 1.316$. Our new algorithm, described below, has reduced both time and storage requirements by virtue of a complexity which is proportional to λ_3^n, where $\lambda_3 \approx 1.20$. It is worth noting that for simpler restricted cases it possible to devise much more efficient algorithms. For problems such as the enumeration of convex polygons [12] and almost-convex polygons [10, 22] the algorithms are of polynomial complexity. Other interesting and related problems for which efficient transfer matrix algorithms can be devised include Hamiltonian circuits on rectangular strips (or other compact shapes) [17] and self-avoiding random walks [6, 4].

In the next section we will very briefly review the finite-lattice method for enumerating square-lattice polygons and give some details of the improved algorithm. The results of the analysis of the series are presented in section 3 including a detailed discussion of a conjecture for the exact critical point.

2. Enumeration of polygons

The method used to enumerate SAP on the square lattice is an enhancement of the method devised by Enting [7] in his pioneering work. The first terms in the series for the polygon-generating function can be calculated using transfer matrix techniques to count the number of polygons in rectangles $W + 1$ edges wide and $L + 1$ edges long. The transfer matrix technique involves drawing a line through the rectangle intersecting a set of $W + 2$ edges. For each configuration of occupied or empty edges along the intersection we maintain a (perimeter) generating function for loops to the left of the line cutting the intersection in that particular pattern. Polygons in a given rectangle are enumerated by moving the intersection so as to add

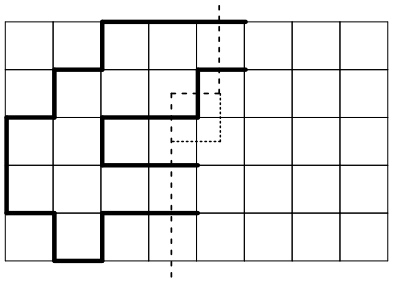

Figure 1. A snapshot of the intersection (dashed line) during the transfer matrix calculation on the square lattice. Polygons are enumerated by successive moves of the kink in the intersection, as exemplified by the position given by the dotted line, so that one vertex at a time is added to the rectangle. To the left of the intersection we have drawn an example of a partially completed polygon.

one vertex at a time, as shown in figure 1. The allowed configurations along the intersection are described in [7]. Each configuration can be represented by an ordered set of edge states $\{n_i\}$, where

$$n_i = \begin{cases} 0 & \text{empty edge} \\ 1 & \text{lower part of loop closed to the left} \\ 2 & \text{upper part of loop closed to the left.} \end{cases} \tag{2}$$

Configurations are read from the bottom to the top. So the configuration along the intersection of the polygon in figure 1 is {0112122}.

The rules for updating the partial generating functions as the intersection is moved are identical to the original work, so we refer the interested reader to [7] for further details regarding this aspect of the transfer matrix calculation.

Due to the obvious symmetry of the lattice one need only consider rectangles with $L \geqslant W$. Valid polygons were required to span the enclosing rectangle in the lengthwise direction. So it is clear that polygons with projection on the y-axis $< W$, that is polygons which are narrower than the width of the rectangle, are counted many times. It is however easy to obtain the polygons of width exactly W and length exactly L from this enumeration [7]. Any polygon spanning such a rectangle has a perimeter of length at least $2(W + L)$. By adding the contributions from all rectangles of width $W \leqslant W_{\max}$ (where the choice of $W_{\max}$ depends on available computational resources, as discussed below) and length $W \leqslant L \leqslant 2W_{\max} - W + 1$, with contributions from rectangles with $L > W$ counted twice, the number of polygons per vertex of an infinite lattice is obtained correctly up to perimeter $4W_{\max} + 2$.

The major improvement of the method used to enumerate polygons in this paper is that we require valid polygons to span the rectangle in *both* directions. In other words we directly enumerate polygons of width exactly W and length L rather than polygons of width $\leqslant W$ and length L as was done originally. The only drawback of this approach is that for most configurations we have to use four distinct generating functions since the partially completed polygon could have reached neither, both, the lower, or the upper boundaries of the rectangle. The major advantage is that the memory requirement of the algorithm is exponentially smaller.

Realizing the full savings in memory usage requires two enhancements to the original algorithm. Firstly, for each configuration we must keep track of the current minimum number of steps N_{cur} that have been inserted to the left of the intersection in order to build up that particular configuration. Secondly, we calculate the minimum number of additional steps N_{add} required to produce a valid polygon. There are three contributions, namely the number of steps required to close the polygon, the number of steps needed (if any) to ensure that the polygon touches both the lower and upper boundary, and finally the number of steps needed (if any) to extend at least W edges in the lengthwise direction. If the sum $N_{\text{cur}} + N_{\text{add}} > 4W_{\max} + 2$ we can discard the partial generating function for that configuration because it will not make a contribution to the polygon count up to the perimeter lengths we are trying to obtain. For

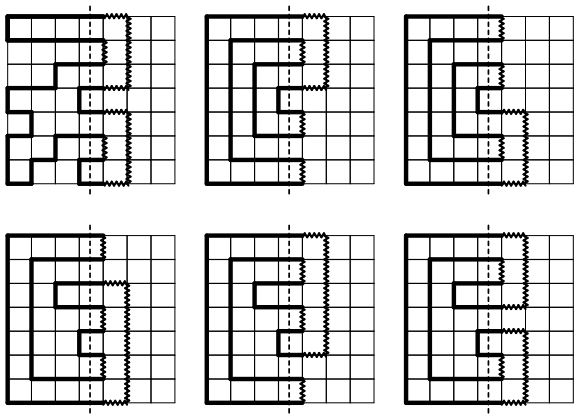

Figure 2. Examples of partially generated polygons (thick solid lines) to the left of the intersection (dashed line) and how to close them in a valid way (thick wavy line). Upper-left panel shows how to close the configuration {12112212}. The upper-middle and right panels show the two possible closures of the configuration {11112222}. The lower panels show the three possible closures of the configuration {11121222}.

instance, polygons spanning a rectangle with a width close to $W_{\max}$ have to be almost convex, so very convoluted polygons are not possible. Thus configurations with many loop ends (non-zero entries) make no contribution at perimeter length $\leqslant 4W_{\max} + 2$.

The number of steps needed to ensure a spanning polygon is straightforward to calculate. The complicated part of the new approach is the algorithm to calculate the number of steps required to close the polygon. There are very many special cases depending on the position of the kink in the intersection and whether or not the partially completed polygon has reached the upper or lower boundary of the bounding rectangle. So in the following we will only briefly describe some of the simple contributions to the closing of a polygon. Firstly, if the partial polygon contains separate pieces these have to be connected as illustrated in figure 2. Separate pieces are easy to locate since all we have to do is start at the bottom of the intersection and moving upwards we count the number of 1's and 2's in the configuration. Whenever these numbers are equal a separate piece has been found and (provided one is not at the last edge in the configuration) the currently encountered 2-edge can be connected to the next 1-edge above. N_{add} is incremented by the number of steps (the distance) between the edges and the two edge-states are removed from the configuration before further processing. It is a little less obvious that if the configuration start (end) as $\{112\ldots2\}$ ($\{1\ldots122\}$) the two lower (upper) edges can safely be connected (note that there can be any number of 0's interspersed before the . . .). Again N_{add} is incremented by the number of steps between the edges, and the two edge-states are removed from the configuration—leading to the new configuration $\{001\ldots2\}$ ($\{1\ldots200\}$)—before further processing. After these operations we may be left with a configuration which has just one 1- and one 2-edge, in which case we are done since these two edges can be connected to form a valid polygon. This is illustrated in figure 2 where the upper-left panel shows how to close the partial polygon with the intersection {12112212}, which contains three separate pieces. After connecting these pieces we are left with the configuration {10012002}. We now connect the two 1-edges and note that the first 2-edge is relabelled to a 1-edge (it has become the new lower end of the loop). Thus we get the configuration {00001002} and we can now connect the remaining two edges and end up with a valid completed polygon. Note that in the last two cases, in addition to the steps spanning the distance between the edges, an additional

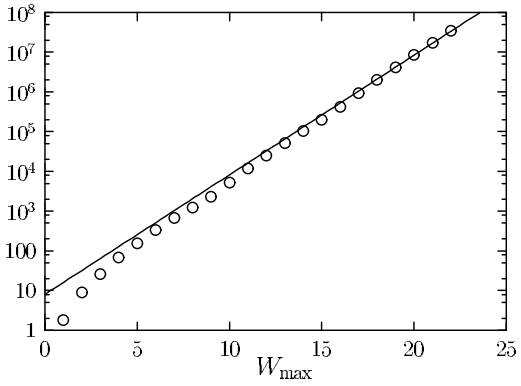

Figure 3. The number of configurations required as $W_{\max}$ is increased. The straight line corresponds to a growth factor $\lambda = 2$.

two horizontal steps had to be added in order to form a valid loop around the intervening edges. If the transformation above does not result in a closed polygon we must have a configuration of the form $\{111\ldots222\}$. The difficulty lies in finding the way to close such configurations with the smallest possible number of additional steps. Suffice to say that if the number of non-zero entries is small one can easily devise an algorithm to try all possible valid ways of closing a polygon and thus find the minimum number of additional steps. In figure 2 we show all possible ways of closing polygons with eight non-zero entries. Note that we have shown the generic cases here. In actual cases there could be any number of 0-edges interspersed in the configurations and this would determine which way of closing would require the least number of additional steps.

With the original algorithm the number of configurations required as $W_{\max}$ increased grew asymptotically as $3^{W_{\max}}$ [11]. Our enumerations indicate that the computational complexity is reduced significantly. While the number of configurations still grows exponentially as $\lambda^{W_{\max}}$ the value of λ is reduced from $\lambda = 3$ to $\lambda \simeq 2$ with the improved algorithm (figure 3 shows the number of configuration required as $W_{\max}$ increases). Furthermore, for any W we know that contributions will start at $4W$ since the smallest polygons have to span a $W \times W$ rectangle. So for each configuration we need only retain $4(W_{\max} - W) + 2$ terms of the generating functions while in the original algorithm contributions started at $2W$ because the polygons were required to span only in the lengthwise direction. We also note that on the square lattice all SAPs are of even length so for each configuration every other term in the generating function is zero, which allows us to discard half the terms and retain only the non-zero ones.

Finally, a few remarks of a more technical nature. The number of contributing configurations becomes very sparse in the total set of possible states along the boundary line and as is standard in such cases one uses a hash-addressing scheme [24]. Since the integer coefficients occurring in the series expansion become very large, the calculation was performed using modular arithmetic [18]. This involves performing the calculation modulo various prime numbers p_i and then reconstructing the full integer coefficients at the end. In order to save memory we used primes of the form $p_i = 2^{15} - r_i$ so that the residues of the coefficients in the polynomials could be stored using 16 bit integers. The Chinese remainder theorem ensures that any integer has a unique representation in terms of residues. If the largest absolute values occurring in the final expansion is m, then we have to use a number of primes k such that $p_1 p_2 \ldots p_k/2 > m$. Up to eight primes were needed to represent the coefficients correctly.

Combining all the memory minimization tricks mentioned above allows us to extend the series for the square-lattice polygon generating function from 70 to 90 terms using at most 2 Gb of memory. Obtaining a series this long with the original algorithm would have required at

Table 1. The number, x_n, of embeddings of n-step polygons on the square lattice. Only non-zero terms are listed.

n	x_n	n	x_n
58	59 270 905 595 010 696 944	76	1 158 018 792 676 190 545 425 711 414
60	379 108 737 793 289 505 364	78	7 554 259 214 694 896 127 239 818 088
62	2 431 560 774 079 622 817 356	80	49 360 379 260 931 646 965 916 677 280
64	15 636 142 410 456 687 798 584	82	323 028 185 951 187 646 733 521 902 740
66	100 792 521 026 456 246 096 640	84	2 117 118 644 744 425 875 029 583 096 670
68	651 206 027 727 607 425 003 232	86	13 895 130 612 692 826 326 409 919 713 700
70	4 216 407 618 470 423 070 733 556	88	91 319 729 650 588 816 198 004 801 698 400
72	27 355 731 801 639 756 123 505 014	90	600 931 442 757 555 468 862 970 353 941 700
74	177 822 806 050 324 126 648 352 460		

Table 2. Estimates for the critical point x_c^2 and exponent $2-\alpha$ obtained from first- and second-order inhomogeneous differential approximants (DA) to the series for square-lattice polygon-generating function. L is the order of the inhomogeneous polynomial.

	First-order DA		Second-order DA	
L	x_c^2	$2-\alpha$	x_c^2	$2-\alpha$
1	0.143 680 628 97(17)	1.500 000 74(35)	0.143 680 628 83(45)	1.500 000 92(92)
2	0.143 680 629 02(14)	1.500 000 68(26)	0.143 680 629 43(29)	1.499 999 57(80)
3	0.143 680 628 78(35)	1.500 001 07(71)	0.143 680 629 14(20)	1.500 000 34(51)
4	0.143 680 629 10(29)	1.500 000 38(61)	0.143 680 629 14(16)	1.500 000 38(44)
5	0.143 680 628 90(43)	1.500 000 85(93)	0.143 680 629 11(53)	1.500 000 2(12)
6	0.143 680 628 63(49)	1.500 001 4(10)	0.143 680 629 01(54)	1.500 000 5(12)
7	0.143 680 628 86(39)	1.500 000 94(80)	0.143 680 628 81(52)	1.500 000 9(10)
8	0.143 680 628 85(64)	1.500 000 8(13)	0.143 680 629 210(97)	1.500 000 21(24)

least 200 times as much memory, or close to half a terabyte! The calculations were performed on an 8-node AlphaServer 8400 with a total of 8 Gb memory. The total CPU time required was about a week per prime. Obviously the calculation for each width and prime are totally independent and several calculations were done simultaneously.

In table 1 we have listed the new terms obtained in this work. They of course agree with the terms up to length 70 computed using the old algorithm. The number of polygons of length $\leqslant 56$ can be found in [11].

3. Analysis of the series

We analysed the series for the polygon generating function by the numerical method of differential approximants [13]. In table 2 we have listed estimates for the critical point x_c^2 and exponent $2-\alpha$ of the series for the square-lattice SAP generating function. The estimates were obtained by averaging values obtained from first-order $[L/N; M]$ and second-order $[L/N; M; K]$ inhomogeneous differential approximants. For each order L of the inhomogeneous polynomial we averaged over those approximants to the series which used at least the first 35 terms of the series (that is, polygons of perimeter at least 74), and used approximants such that the difference between N, M, and K did not exceed two. These are therefore 'diagonal' approximants. Some approximants were excluded from the averages because the estimates were obviously spurious. The error quoted for these estimates reflects

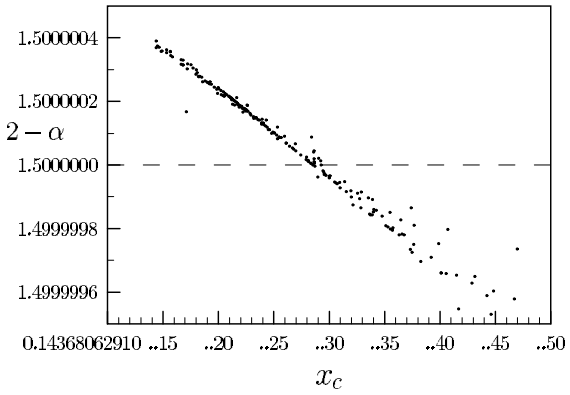

Figure 4. Estimates for the critical exponent $2-\alpha$ versus estimates for the critical point x_c^2 of the square-lattice polygon-generating function.

the spread (basically one standard deviation) among the approximants. Note that these error bounds should *not* be viewed as a measure of the true error as they cannot include possible systematic sources of error. We discuss further the systematic error when we consider biased approximants. Based on these estimates we conclude that $x_c^2 = 0.143\,680\,6289(5)$ and $\alpha = 0.500\,0005(10)$.

As stated earlier there is very convincing evidence that the critical exponent $\alpha = \frac{1}{2}$ exactly. If we assume this to be true we can obtain a refined estimate for the critical point x_c^2. In figure 4 we have plotted estimates for the critical exponent $2-\alpha$ against estimates for the critical point x_c^2. Each dot in this figure represents a pair of estimates obtained from a second-order inhomogeneous differential approximant. The order of the inhomogeneous polynomial was varied from 0 to 10. We observe that there is an almost linear relationship between the estimates for $2-\alpha$ and x_c^2 and that for $2-\alpha = \frac{3}{2}$ we get $x_c^2 \simeq 0.14\,368\,062\,928\ldots$. In order to get some idea as to the effect of systematic errors, we carried out this analysis using polygons of length up to 60 steps, then 70, then 80 and finally 90 steps. The results were $x_c^2 = 0.143\,680\,6308$ for $n = 60$, $x_c^2 = 0.143\,680\,629\,56$ for $n = 70$, $x_c^2 = 0.143\,680\,629\,30$ for $n = 80$, and $x_c^2 = 0.143\,680\,629\,28$ for $n = 90$. This is a rapidly converging sequence of estimates, though we have no theoretical basis that would enable us to assume any particular rate of convergence. However, observing that the differences between successive estimates are decreasing by a factor of at least five, it is not unreasonably optimistic to estimate the limit at $x_c^2 = 0.143\,680\,629\,27(1)$.

This leads to our final estimate $x_c^2 = 0.143\,680\,629\,27(1)$ and thus we find the connective constant $\mu = 1/x_c = 2.638\,158\,530\,34(10)$. It is interesting to note that some years ago we [3] pointed out that since the hexagonal lattice connective constant is given by the zero of a quadratic in x^2, it is plausible that this might be the case also for the square-lattice connective constant. On the basis of an estimate of the connective constant that was four orders of magnitude less precise, we pointed out then that the polynomial

$$581x^4 + 7x^2 - 13 = 0$$

was the only polynomial we could find with 'small' integer coefficients consistent with our estimate. The relevant zero of this polynomial is $x_c^2 = 0.143\,680\,629\,269\,8685\ldots$ in complete agreement with our new estimate—which, as noted above, contains four more significant digits! Unfortunately the other zero is at $x_c^2 = -0.155\,7288\ldots$, and we see no evidence of such a singularity. Nevertheless, the agreement is so astonishingly good that

we are happy to take this as a good algebraic approximation to the connective constant. An argument as to why we might not expect to see the singularity on the negative real axis from our series analysis would make the root of the above polynomial a plausible conjecture for the exact value, but at present such an argument is missing.

Two further analyses were carried out on the data. Firstly, a study of the location of non-physical singularities, and secondly, a study of the asymptotic form of the coefficients—which is relevant to the identification of any correction-to-scaling exponent. Singularities outside the radius of convergence give exponentially small contributions to the asymptotic form of the coefficients, so are notoriously hard to analyse. Nevertheless, we see clear evidence of a singularity on the negative real axis at $x^2 \approx -0.40$ with an exponent that is extremely difficult to analyse but could be 1.5, in agreement with the physical exponent. There is weaker evidence of a further conjugate pair of singularities. First-order approximants locate these at -0.015 ± 0.36i, while second-order approximants locate them at -0.035 ± 0.31i. There is also evidence of a further singularity on the negative real axis at $x_c^2 = -0.7$. We are unable to give a useful estimate of the exponents of these singularities.

We turn now to the asymptotic form of the coefficients. We have argued previously [4] that there is no non-analytic correction-to-scaling exponent for the polygon generating function. This is entirely consistent with Nienhuis's [23] observation that there is a correction-to-scaling exponent of $\Delta = \frac{3}{2}$. Since for the polygon-generating function exponent $\alpha = \frac{1}{2}$, the correction term has an exponent equal to a positive integer, and therefore 'folds into' the analytic background term, denoted $A(x)$ in equation (1). This is explained in greater detail in [4]. We assert that the asymptotic form for the polygon-generating function is as given by equation (1) above. In evidence of this, we remark that from (1) follows the asymptotic form

$$p_{2n} x_c^{2n} \sim n^{-\frac{5}{2}} [a_1 + a_2/n + a_3/n^2 + a_4/n^3 + \cdots]. \tag{3}$$

Using our algebraic approximation to x_c quoted above, we show in table 3 the estimates of the amplitudes $a_1, \ldots, a_4$. From the table we see that $a_1 \approx 0.099\,4018$, $a_2 \approx -0.027\,51$, $a_3 \approx 0.0255$ and $a_4 \approx 0.12$, where in all cases we expect the error to be confined to the last quoted digit. The excellent convergence of all columns is strong evidence that the assumed asymptotic form is correct. If we were missing a term corresponding to, say, a half-integer correction, the fit would be far worse. This is explained at greater length in [4]. So good is the fit to the data that if we take the last entry in the table, corresponding to $n = 45$, and use the entries as the amplitudes, then $p_4 \ldots p_{16}$ are given exactly by the above asymptotic form (provided we round to the nearest integer), and beyond perimeter 20 all coefficients are given to the same accuracy as the leading amplitude.

Finally, to complete our analysis, we estimate the critical amplitudes $A(x_c^2)$ and $B(x_c^2)$, defined in equation (1). $A(x_c^2)$ has been estimated by evaluating Padé approximants to the generating function, evaluated at x_c^2. In this way we estimate $A(x_c^2) \approx 0.036$, while $B(x_c^2)$ follows from the estimate of a_1 in equation (3), since $B(x_c^2) = \frac{4\sqrt{\pi} a_1}{3} \approx 0.234\,913$.

4. Conclusion

We have presented an improved algorithm for the enumeration of SAPs on the square lattice. The computational complexity of the algorithm is estimated to be 1.2^n. Implementing this algorithm has enabled us to obtain polygons up to perimeter length 90. Decomposing the coefficients into prime factors reveals frequent occurrence of very large prime factors, supporting the widely held view that there is no 'simple' formula for the coefficients. For example, p_{78} contains the prime factor 7789 597 345 683 901 619. Our extended series enables us to give an extremely precise estimate of the connective constant, and we give a simple

Table 3. A fit to the asymptotic form $p_{2n}x_c^{2n} \sim n^{-\frac{5}{2}}[a_1 + a_2/n + a_3/n^2 + a_4/n^3 + \cdots]$. Estimates of the amplitudes a_1, a_2, a_3, a_4.

n	a_1	a_2	a_3	a_4
20	0.099 400 85	−0.027 457 05	0.024 763 76	0.118 221 81
21	0.099 401 18	−0.027 475 48	0.025 113 47	0.116 011 07
22	0.099 401 40	−0.027 488 80	0.025 379 79	0.114 238 55
23	0.099 401 54	−0.027 497 67	0.025 565 92	0.112 937 66
24	0.099 401 64	−0.027 503 97	0.025 704 26	0.111 924 57
25	0.099 401 70	−0.027 508 29	0.025 803 64	0.111 163 57
26	0.099 401 74	−0.027 511 37	0.025 877 57	0.110 572 83
27	0.099 401 77	−0.027 513 55	0.025 932 11	0.110 118 80
28	0.099 401 79	−0.027 515 10	0.025 972 36	0.109 770 30
29	0.099 401 80	−0.027 516 19	0.026 001 68	0.109 506 67
30	0.099 401 81	−0.027 516 94	0.026 022 73	0.109 310 43
31	0.099 401 82	−0.027 517 45	0.026 037 34	0.109 169 29
32	0.099 401 82	−0.027 517 77	0.026 046 92	0.109 073 54
33	0.099 401 82	−0.027 517 95	0.026 052 54	0.109 015 52
34	0.099 401 82	−0.027 518 02	0.026 055 00	0.108 989 29
35	0.099 401 82	−0.027 518 02	0.026 054 94	0.108 989 93
36	0.099 401 82	−0.027 517 96	0.026 052 85	0.109 013 58
37	0.099 401 82	−0.027 517 85	0.026 049 13	0.109 056 99
38	0.099 401 82	−0.027 517 71	0.026 044 08	0.109 117 57
39	0.099 401 82	−0.027 517 55	0.026 037 96	0.109 193 02
40	0.099 401 82	−0.027 517 36	0.026 030 97	0.109 281 58
41	0.099 401 82	−0.027 517 17	0.026 023 27	0.109 381 60
42	0.099 401 81	−0.027 516 96	0.026 015 00	0.109 491 74
43	0.099 401 81	−0.027 516 75	0.026 006 29	0.109 610 79
44	0.099 401 81	−0.027 516 53	0.025 997 20	0.109 737 96
45	0.099 401 81	−0.027 516 31	0.025 987 85	0.109 871 95

algebraic approximation that agrees precisely with our numerical estimate. An alternative analysis provides very strong evidence for the absence of any non-analytic correction terms to the proposed asymptotic form for the generating function. Finally, we give an asymptotic representation for the coefficients which we believe to be accurate for all positive integers.

Acknowledgments

We gratefully acknowledge valuable discussions with Ian Enting, useful comments on the manuscript by Alan Sokal and financial support from the Australian Research Council.

References

[1] Baxter R J 1986 *J. Phys. A: Math. Gen.* **19** 2821
[2] Brak R and Guttmann A J 1990 *J. Phys. A: Math. Gen.* **23** 4581
[3] Conway A R, Enting I G and Guttmann A J 1993 *J. Phys. A: Math. Gen.* **26** 1519
[4] Conway A R and Guttmann A J 1996 *Phys. Rev. Lett.* **77** 5284
[5] Delest M P and Viennot G 1984 *Theor. Comput. Sci.* **34** 169
[6] Derrida B 1981 *J. Phys. A: Math. Gen.* **14** L5
[7] Enting I G 1980 *J. Phys. A: Math. Gen.* **13** 3713
[8] Enting I G and Guttmann A J 1985 *J. Phys. A: Math. Gen.* **18** 1007
[9] Enting I G and Guttmann A J 1989 *J. Phys. A: Math. Gen.* **22** 1371
[10] Enting I G, Guttmann A J, Richmond L B and Wormald N C 1992 *Random Structures and Algorithms* **3** 445

[11] Guttmann A J and Enting I G 1988 *J. Phys. A: Math. Gen.* **21** L165
[12] Guttmann A J and Enting I G 1988 *J. Phys. A: Math. Gen.* **21** L467
[13] Guttmann A J 1989 *Phase Transitions and Critical Phenomena* vol 13, ed C Domb and J L Lebowitz (New York: Academic)
[14] Hughes B D 1995 *Random Walks and Random Environments, Vol I Random Walks* (Oxford: Clarendon)
[15] Jensen I and Guttmann A J 1998 *J. Phys. A: Math. Gen.* **31** 8137
[16] Kim D 1988 *Discrete Math.* **70** 47
[17] Kloczkowski A and Jernigan R L 1998 *J. Chem. Phys.* **109** 5134
Kloczkowski A and Jernigan R L 1998 *J. Chem. Phys.* **109** 5147
[18] Knuth D E 1969 *Seminumerical Algorithms (The Art of Computer Programming vol 2)* (Reading, MA: Addison-Wesley)
[19] Lin K Y and Chang S J 1988 *J. Phys. A: Math. Gen.* **21** 2635
[20] Lin K Y and Tzeng W J 1991 *Int. J. Mod. Phys.* B **5** 1913
Tzeng W J and Lin K Y 1991 *Int. J. Mod. Phys.* B **5** 2551
[21] Lin K Y 1992 *J. Phys. A: Math. Gen.* **25** 1835
[22] Moraal H 1994 *Physica* A **203** 91
Moraal H 1994 *Physica* A **203** 103
[23] Nienhuis B 1982 *Phys. Rev. Lett.* **49** 1062
[24] Mehlhorn K 1984 *Data Structures and Algorithms I: Sorting and Searching (EATCS Monographs on Theoretical Computer Science)* (Berlin: Springer)
[25] Pólya G 1969 *J. Comb. Theory* **6** 102
[26] Temperley H N V 1956 *Phys. Rev.* **103** 1