Calculation of the connective constant for self-avoiding walks via the pivot algorithm

IOP Publishing — Journal of Physics A: Mathematical and Theoretical

J. Phys. A: Math. Theor. **46** (2013) 245001 (11pp) doi:10.1088/1751-8113/46/24/245001

Calculation of the connective constant for self-avoiding walks via the pivot algorithm

Nathan Clisby

ARC Centre of Excellence for Mathematics and Statistics of Complex Systems,
Department of Mathematics and Statistics, The University of Melbourne, VIC 3010, Australia

E-mail: n.clisby@ms.unimelb.edu.au

Received 1 March 2013, in final form 4 May 2013
Published 24 May 2013
Online at stacks.iop.org/JPhysA/46/245001

Abstract
We calculate the connective constant for self-avoiding walks on the simple cubic lattice to unprecedented accuracy, using a novel application of the pivot algorithm. We estimate that $\mu = 4.684\,039\,931 \pm 0.000\,000\,027$. Our method also provides accurate estimates of the number of self-avoiding walks, even for walks with millions of steps.

PACS numbers: 05.50.+q, 05.10.−a, 05.70.Jk, 02.10.Ox
Mathematics Subject Classification: 82B20, 82B41, 82B80

1. Introduction

The self-avoiding walk (SAW) on a regular lattice is an important model in statistical mechanics with a long history [1]. An N-step SAW is a map ω from the integers $\{0, 1, \ldots, N\}$ to sites on the lattice, with $\omega(0)$ conventionally at the origin, $|\omega(i+1) - \omega(i)| = 1$, and $\omega(i) \neq \omega(j)\ \forall i \neq j$. SAW is a topic of much current interest: see [2] for a recent review of rigorous results, and [3] for an overview of self-avoiding polygons (SAPs) which has broader scope, including numerical aspects of SAP and to a lesser extent SAW.

The most important quantities which characterize SAW are the number of SAW of length N, c_N, and measures of the size of the walk, such as the square end-to-end distance. The asymptotic behavior of c_N on the simple cubic lattice is believed to be

$$c_N \sim A\mu^N N^{\gamma-1}(1 + O(N^{-\Delta_1})), \tag{1.1}$$

where the connective constant μ and amplitude A are lattice dependent, the critical exponent γ is universal, and Δ_1 is the exponent of the leading correction to scaling. There are also sub-leading analytic corrections to scaling, and a contribution from the so-called anti-ferromagnetic singularity; see for example [4] and [5] for more details on the asymptotic form of c_N in two and three dimensions respectively.

Enumeration is a particularly powerful method for studying SAW on two-dimensional lattices, where the finite lattice method is highly effective [6–8]. The best estimate for μ on the square lattice comes from enumerations of SAPs to 130 steps [9], leading to the highly accurate estimate $\mu = 2.638\,158\,530\,35(2)$. For the simple cubic lattice, the best estimate for μ comes from pruned-enriched Rosenbluth method (PERM) Monte Carlo simulations [10]: $\mu = 4.684\,0386(11)$. The most powerful known enumeration method for three-dimensional lattices is the length-doubling algorithm [11], which combines brute force enumeration with the inclusion–exclusion principle in a novel way. SAW on the simple cubic lattice have been enumerated to 36 steps, with $c_{36} = 2\,941\,370\,856\,334\,701\,726\,560\,670$, and $\mu = 4.684\,0401(50)$ [11].

In this paper we will obtain a highly accurate estimate of μ for SAW on the simple cubic lattice using a Monte Carlo algorithm. Our method can also be used to estimate the number of SAWs.

2. Method

Our method to calculate μ for SAW combines four key ideas.

(i) Use of the pivot algorithm, the most powerful known method for sampling SAW.
(ii) A novel computer experiment which involves a telescoping sum that eliminates corrections to scaling.
(iii) The adoption of scale-free moves to efficiently calculate the observable of interest.
(iv) Partitioning CPU time between different sub-problems in an optimal way.

We now describe each of these aspects in turn.

2.1. The pivot algorithm

The pivot algorithm is an extremely powerful method for sampling SAW in the canonical ensemble. It was invented by Lal [12], but the true power of the method was only appreciated after the ground-breaking work of Madras and Sokal [13]. Recently, the implementation of the pivot algorithm has been improved to make it even more powerful [14–16]. The recent improvements make it an extremely attractive prospect to utilize the pivot algorithm whenever possible.

The pivot algorithm is a Markov chain Monte Carlo algorithm which works in the set of SAWs of fixed length, where the elementary move is a *pivot* as described below. The pivot algorithm generates a correlated sequence of SAW via the following process.

(i) Select a pivot site of the current SAW according to some prescription—usually uniformly at random.
(ii) Randomly choose a lattice symmetry (rotation or reflection).
(iii) Apply this symmetry to one of the two sub-walks created by splitting the walk at the pivot site.
(iv) If the resulting walk is self-avoiding: *accept* the pivot and update the configuration.
(v) If the resulting walk is not self-avoiding: *reject* the pivot and keep the old configuration.
(vi) Repeat.

The pivot algorithm is ergodic, and satisfies the detailed balance condition which ensures that SAW are sampled uniformly at random [13].

After a successful pivot, *global* observables, such as the square end-to-end distance, change significantly and are essentially uncorrelated. This observation is equivalent to the statement that the integrated autocorrelation time for a global observable A, $\tau_{\text{int}}(A)$, is of the same order as the mean time for a successful pivot. In the language of [13], once a successful pivot has been made the resulting configuration is 'essentially new' with respect to global observables. For SAW on the simple cubic lattice the probability of a pivot attempt being

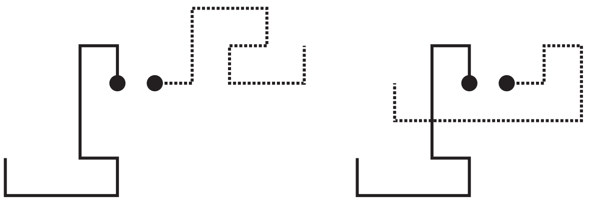

Figure 1. Concatenation of two walks on the square lattice. On the left the indicator function $B(\omega_1, \omega_2) = 1$, while on the right $B(\omega_1, \omega_2) = 0$.

successful is $O(N^{-p})$, with $p \approx 0.11$. Therefore global observables have $\tau_{\text{int}} = O(N^p)$; see [13] for extensive discussion.

For *local* observables, such as the angle between the 37th and 38th steps of a walk, one may need $O(N)$ successful pivots before the observable changes. Consequently $\tau_{\text{int}} = O(N^{1+p})$ for local observables.

2.2. Telescoping observable

Given walks ω_1 and ω_2, we define a concatenation operation by placing the root point of ω_1 at the origin, and the root point of ω_2 at $(1, 0, 0)$. We denote the resulting walk as $\omega_1 \circ \omega_2$. Under this definition of concatenation, walks of M and N steps are fused together to create a walk of $M + N + 1$ steps. We now define the observable of interest to be the indicator function defined as follows:

$$B(\omega_1, \omega_2) = \begin{cases} 0 & \text{if } \omega_1 \circ \omega_2 \text{ not self-avoiding} \\ 1 & \text{if } \omega_1 \circ \omega_2 \text{ self-avoiding.} \end{cases} \tag{2.1}$$

See figure 1 for two examples of concatenation.

The more common definition for concatenation has the root points for the two walks placed at the origin. We use an alternate definition because it is straightforward to calculate the indicator function using our SAW-tree implementation [16].

If we let Ω be the coordination number of the lattice ($\Omega = 6$ for the simple cubic lattice), we then have

$$B_{M,N} \equiv \text{Mean value of } B(\omega_1, \omega_2) \text{ over all pairs of } M \text{ and } N \text{ step walks,} \tag{2.2}$$

$$= \langle B(\omega_1, \omega_2)\rangle_{|\omega_1|=M, |\omega_2|=N}, \tag{2.3}$$

$$= \frac{1}{c_M c_N} \sum_{|\omega_1|=M, |\omega_2|=N} B(\omega_1, \omega_2), \tag{2.4}$$

$$= \frac{c_{M+N+1}}{\Omega c_M c_N}. \tag{2.5}$$

The longest walks which have been exactly enumerated on the simple cubic lattice have 36 steps [11], and we can recursively exploit this fact. For convenience we define

$$\widetilde{B}_N \equiv \Omega B_{N,N} = \frac{c_{2N+1}}{c_N^2}, \tag{2.6}$$

and so

$$c_{73} = \widetilde{B}_{36} c_{36}^2, \tag{2.7}$$

$$c_{147} = \widetilde{B}_{73} c_{73}^2 = \widetilde{B}_{73} \widetilde{B}_{36}^2 c_{36}^4, \tag{2.8}$$

$$c_{295} = \widetilde{B}_{147}\widetilde{B}_{73}^{2}\widetilde{B}_{36}^{4}c_{36}^{8}, \tag{2.9}$$

$$\vdots$$

$$c_{38\,797\,311} = \widetilde{B}_{19\,398\,655}\widetilde{B}_{9699\,327}^{2}\cdots\widetilde{B}_{36}^{2^{19}}c_{36}^{2^{20}}. \tag{2.10}$$

Thus, estimates for $\widetilde{B}_N$ can be mapped to estimates of the number of walks c_N. We can then use equation (1.1) to estimate μ:

$$\mu_N \equiv c_N^{1/N} \tag{2.11}$$

$$\therefore \log \mu_N = \frac{1}{N}\log c_N \tag{2.12}$$

$$\sim \log\mu + \frac{(\gamma-1)\log N}{N} + \frac{\log A}{N} + O(N^{-\Delta_1 - 1}). \tag{2.13}$$

Corrections to scaling vanish with increasing N, and estimates for μ_N approach μ.

Taking the logarithm of each side of equations (2.7)–(2.10), one can see that the contribution of the c_{36} term remains approximately constant, but the addition of higher order terms successively eliminate the higher order corrections. In particular,

$$\log\mu_{38\,797\,311} = \frac{1}{38\,797\,311}\log\widetilde{B}_{19\,398\,655} + \frac{2}{38\,797\,311}\log\widetilde{B}_{9699\,327} \cdots + \frac{2^{19}}{38\,797\,311}\log\widetilde{B}_{36} + \frac{2^{20}}{38\,797\,311}\log c_{36}. \tag{2.14}$$

The approach described here may be thought of as a 'divide-and-conquer' algorithm, where a long SAW is successively split into halves. This is in stark contrast to typical growth algorithms such as PERM, where SAW (and other combinatorial objects) are incrementally built up step by step.

2.3. Scale-free moves

In order to accurately estimate μ from equation (2.14), we must find an efficient way to estimate $\widetilde{B}_N$. We estimate $\widetilde{B}_N$ by sampling pairs of SAW of length N via the pivot algorithm, and then $\widetilde{B}_N$ is the time average of $\Omega B(\omega_1, \omega_2)$. The observable B is *not* a global observable in the same sense as, for example, the square end-to-end distance: it clearly depends strongly on the details of the structure of each walk close to the concatenation joint.

We now present a simple yet subtle argument to show that if we naively sample pivot sites uniformly at random, then τ_{int} for B will be $O(N)$. We will assume throughout that we are considering pairs of walks of length N.

First, let us define zero atmosphere SAW as those SAWs for which one of the ends has all neighboring sites occupied. It is well known that zero atmosphere walks have positive density in the set of all walks (see e.g. [17]). We denote a zero atmosphere SAW as 'minimal' if, starting from the end, we visit all of the neighbors of the end in the fewest possible number of steps. Minimal zero atmosphere walks also have positive density in the set of SAW. E.g. for the square lattice, the density of minimal zero atmosphere SAW which start with the seven steps in figure 2 is bounded below as the SAW length $N \to \infty$.

Our ensemble is pairs of SAW, each of fixed length. Suppose we were to sample pivot sites uniformly at random, so generating a Markov chain. Assume we have equilibrated the Markov chain so that we are guaranteed to be sampling from the equilibrium distribution. If

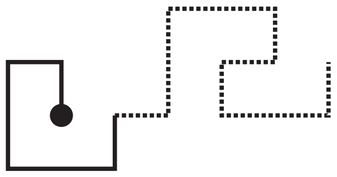

Figure 2. Minimal trapped walk of seven steps on the square lattice (solid line) with a possible extension (dashed line).

we were to choose a random time in the Markov chain, the probability of choosing a minimal zero atmosphere walk is then $O(1)$. However, the probability that the next pivot site chosen could change the value of the atmosphere is $O(1/N)$. Therefore, in this case B will, on average, remain zero for $O(N)$ time steps in the Markov chain. For the observable B, the contribution of zero atmosphere walks ensures that it must take time $O(N)$ to achieve an essentially new configuration. Thus, $\tau_{\mathrm{int}}(B) = O(N)$ when pivots are sampled uniformly at random. Note that this effect is actually quite subtle, as although zero atmosphere walks have positive density, in practice this density is small. Thus the contribution of these configurations to $\tau_{\mathrm{int}}(B)$ is small in practice until N is of the order of thousands or tens of thousands.

However, it is possible to dramatically improve the integrated autocorrelation time for B, and hence the accuracy of our estimate of $\widetilde{B}_N$. The key point is that the concatenation operation introduces a new, important length scale into the system, namely the distance from the concatenation joint to internal sites of the walk. B depends strongly on the structure of the walk according to this distance. We make the following conjecture.

Conjecture 1. *Suppose we have an observable for a polymer system that depends on a single internal distance, L. Then the integrated autocorrelation time for this observable is of the same order as the time it takes to make successful pivots at all length scales with respect to this distance.*

To be concrete, if L is the distance from an internal site to the concatenation joint, then we believe that an essentially new configuration with respect to B is obtained once pivots have been made at length scales L of order $1, 2, 4, 8, 16, \ldots, N$.

By choosing pivot sites uniformly at random with respect to $\log L$, we therefore expect that there is only at most a $\log N$ penalty for the integrated autocorrelation time for $\widetilde{B}_N$ as compared to a global observable, i.e. $\tau_{\mathrm{int}}(B) = O(N^p \log N)$. N.B., since the CPU time per attempted pivot for the SAW-tree implementation is $O(\log N)$ [16], this means that in CPU units $\widetilde{\tau}_{\mathrm{int}}(B) = O(N^p \log^2 N)$.

2.4. Experimental design

To estimate μ we must calculate each of the terms in equation (2.14). We do so by running separate Monte Carlo simulations for pairs of walks of length 36, 73, ..., 19 398 655, in order to calculate $\widetilde{B}_N$. Since it takes CPU time $O(\log N)$ to make a pivot attempt, and CPU time $O(\log N)$ to calculate B, we choose to sample B for every time step in the Markov chain. The procedure we used was.

(i) Use the pseudo dimerize procedure of [16] to generate two initial N-step SAW configurations.

(ii) Initialize Markov chain by performing at least $20N$ successful pivots on each SAW. Pivot sites are sampled uniformly at random. The stopping criterion must be based on the number of attempted pivots so as not to introduce bias.

Our sampling procedure for B is then:

(i) Select one of the two walks uniformly at random.
(ii) Select a pivot site on this walk by generating a pseudorandom number x between 0 and $\log N$, and let pivot site $j = \lfloor e^x \rfloor$.
(iii) Attempt pivot move, update walk if result is self-avoiding.
(iv) Randomly pivot each of the walks around their root points. These pivots are always successful.
(v) Calculate $B(\omega_1, \omega_2)$, and update our estimate of $\widetilde{B}_N$.
(vi) Repeat.

Our goal is to optimally partition CPU time amongst the terms in equation (2.14), in order to minimize the overall error in our estimate of μ. The terms in equation (2.14) approach $\frac{2}{N}\log \widetilde{B}_N$ for large N. We have

$$\widetilde{B}_N = \frac{c_{2N+1}}{c_N^2} \sim \frac{A\mu^{2N+1}(2N+1)^{\gamma-1}}{A^2\mu^{2N}N^{2(\gamma-1)}} \sim CN^{1-\gamma}, \tag{2.15}$$

$$\therefore \frac{1}{N}\log \widetilde{B}_N \sim \frac{1-\gamma}{N}\log N + O(1/N). \tag{2.16}$$

The $1/N$ factor on the right hand side of the above equation dominates the increase in integrated autocorrelation time in CPU units for B. Therefore if we were to invest the same CPU time in each term of equation (2.14), the contributions to the error would diminish with increasing N!

To minimize overall statistical error we now perform a short test run of CPU time t_0 for each length, determining the constants a_N in

$$\sigma\left(\frac{1}{N}\log \widetilde{B}_N\right) = \frac{a_N}{\sqrt{t_0}}. \tag{2.17}$$

We show these measured values of a_N in figure 3. However, we can also express σ in terms of the variance of B and the integrated autocorrelation time of the algorithm. Assuming that conjecture 1 is correct, modulo logarithmic factors we obtain the following expression for a_N:

$$a_N \sim N^{-1+(p+\gamma-1)/2} \approx N^{-0.87}. \tag{2.18}$$

In figure 3, it is clear that a_N decays as a power law with N, as expected. By inspection, a_N follows the predicted power law behavior quite closely, and thus figure 3 provides strong numerical support for conjecture 1.

We then fix the total running time for our computer experiment at t. The (statistical) square error in our estimate for μ is then

$$\sigma^2 = \sum \frac{a_i^2}{t_i}, \quad \text{subject to } t = \sum t_i. \tag{2.19}$$

The optimal choice of t_i to minimize σ^2 is then

$$t_i = \frac{a_i}{\sum a_i} t, \tag{2.20}$$

and the optimal value for the error is

$$\sigma = \frac{\sum a_i}{\sqrt{t}}. \tag{2.21}$$

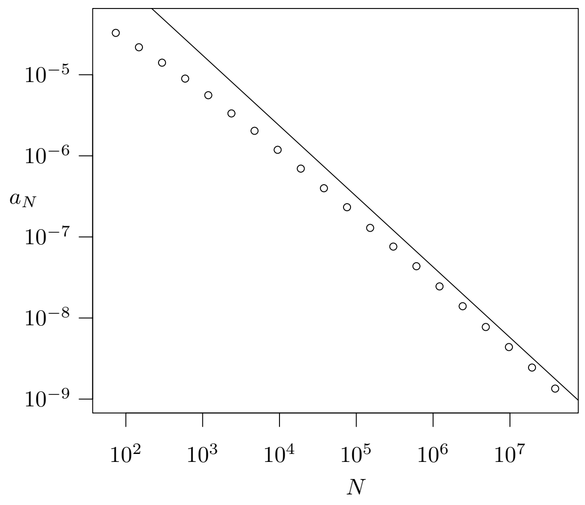

Figure 3. Measured values of a_N, which measures the expected error of contributions to equation (2.14), in units of $\sqrt{\text{seconds}}$. A line of slope $(-1 + (p + \gamma - 1)/2) \approx -0.87$ is included in the plot for comparison.

In practice, we did not rigorously apply this prescription to the longest walks, and instead spent at minimum 1% of the CPU time at each length.

Almost all of the computational effort is spent on the $\widetilde{B}_{36}$ and $\widetilde{B}_{73}$ terms in equation (2.14). The higher order terms reduce the corrections to scaling, and essentially eliminate the systematic error in our estimate of μ.

3. Results and analysis

The analysis for this computer experiment is remarkably simple. It is an extremely rare example of a problem in lattice statistical mechanics for which we have strong evidence that systematic errors are negligible. Hence the confidence intervals we report are purely statistical.

In table 1 we report our estimates for $\widetilde{B}_N$, and thence our estimates for c_N from equations (2.7)–(2.10). Note that the estimates for c_N are highly correlated. The error in the mantissa is given in the final column; for example, from table 1 we estimate that $c_{38\,797\,311} = 6.6 \times 10^{26\,018\,276}$, with the confidence interval of the mantissa being (5.3, 8.2). This is a direct estimate from our $\widetilde{B}_N$ values: it is *not* an extrapolation, and the reported error is purely statistical. These estimates are sufficiently accurate that we are capable of reliably distinguishing between values of c_N for consecutive values of N, for N in the full range encompassed by table 1, to the 2σ level or better. That is, we can give the precise answer to a question such as: 'if $c_N = 3.\cdots \times 10^{6504\,569}$, then how long must the SAW be?'[1].

We ran the computer experiment for a total of 60 000 CPU hours on SunFire X4600M2 machines with 2.3 GHz AMD Opteron CPUs. In table 1 we record the number of batches of 10^9 Markov chain time steps which were used to generate our data. There were a grand total of 1117 583 batches, or approximately 10^{15} total Markov chain time steps, which took mean CPU time of 0.19 μs. See table 4 of [16] for timings of the SAW-tree implementation for the pivot algorithm where pivot sites are sampled uniformly at random. One can see that

[1] Answer: $N = 9699\,327$.

Table 1. Estimates of $\widetilde{B}_N$ and c_N with statistical errors, and the number of batches of 10^9 Markov chain time steps used to generate these estimates.

N	Batches	$\widetilde{B}_{(N-1)/2}$	c_N	c_N mantissa interval
73	495 002	2.472 670 30(65)	$2.139\,271 \times 10^{49}$	(2.139 270, 2.139 271)
147	287 330	2.207 539 77(91)	$1.010\,276 \times 10^{99}$	(1.010 275, 1.010 277)
295	136 314	1.974 0142(14)	$2.014\,793 \times 10^{198}$	(2.014 790, 2.014 796)
591	86 405	1.766 8271(18)	$7.172\,241 \times 10^{396}$	(7.172 218, 7.172 264)
1183	47 444	1.582 3991(25)	$8.140\,025 \times 10^{793}$	(8.139 971, 8.140 078)
2367	23 351	1.417 8577(36)	$9.394\,724 \times 10^{1587}$	(9.394 599, 9.394 850)
4735	9351	1.270 8081(58)	$1.121\,626 \times 10^{3176}$	(1.121 595, 1.121 656)
9471	4793	1.139 2521(81)	$1.433\,230 \times 10^{6352}$	(1.433 151, 1.433 308)
18 943	3708	1.021 4669(91)	$2.098\,243 \times 10^{12\,704}$	(2.098 013, 2.098 474)
37 887	3663	0.915 9517(92)	$4.032\,592 \times 10^{25\,408}$	(4.031 706, 4.033 477)
75 775	3067	0.821 4372(97)	$1.335\,804 \times 10^{50\,817}$	(1.335 217, 1.336 391)
151 551	2760	0.736 643(10)	$1.314\,444 \times 10^{101\,634}$	(1.313 290, 1.315 600)
303 103	2472	0.660 651(10)	$1.141\,449 \times 10^{203\,268}$	(1.139 445, 1.143 457)
606 207	2443	0.592 531(11)	$7.720\,126 \times 10^{406\,535}$	(7.693 038, 7.747 310)
1212 415	1959	0.531 449(11)	$3.167\,451 \times 10^{813\,071}$	(3.145 262, 3.189 797)
2424 831	1811	0.476 654(11)	$4.782\,146 \times 10^{1626\,142}$	(4.715 379, 4.849 858)
4849 663	1628	0.427 497(11)	$9.776\,394 \times 10^{3252\,284}$	(9.505 309, 1.005 521)
9699 327	1481	0.383 408(12)	$3.664\,531 \times 10^{6504\,569}$	(3.464 124, 3.876 531)
19 398 655	1366	0.343 919(12)	$4.618\,409 \times 10^{13\,009\,138}$	(4.127 077, 5.168 235)
38 797 311	1235	0.308 455(11)	$6.579\,250 \times 10^{26\,018\,276}$	(5.253 839, 8.239 029)

for our SAW-tree data structure the biased pivot site selection scheme utilized in this paper is significantly faster than the usual circumstance where pivots are sampled uniformly at random.

As a technical aside, we note that the error estimates for $\widetilde{B}_N$ in table 1 are approximately constant for $N \geqslant 151\,551$ because we invested the same percentage of CPU time in each of these cases.

In our analysis for μ we utilize an estimate for the critical exponent γ from a companion Monte Carlo computer experiment [18]: $\gamma = 1.156\,96(1)$. In addition, we utilize the estimate of the critical amplitude $A = 1.215(2)$ from [5]. We do this by setting $\gamma^* = 1.156\,96$, $A^* = 1.215$, and forming the improved estimates

$$\log \mu_N^* = \log \mu_N - \frac{(\gamma^* - 1)\log N}{N} - \frac{\log A^*}{N}. \tag{3.1}$$

We denote the errors in the utilized estimates as $\sigma_\gamma = 0.000\,01$ and $\sigma_A = 0.002$. In the limit of large N, μ_N^* will then have the following contributions to the systematic error:

$$\frac{\mu\sigma_\gamma \log N}{N}, \qquad \frac{\mu\sigma_A}{AN}, \qquad O(N^{-\Delta_1-1}). \tag{3.2}$$

The Δ_1 term comes from the leading order correction in equation (2.13). From [15] we have $\Delta_1 = 0.528(12)$. The constant of this term is indeterminate, but we will see that it cannot be so large so as to interfere with our estimates.

Our estimates for μ are collected in table 2. The μ_N^* estimates rapidly converge with increasing N, which indicates that for the largest values of N systematic errors are negligible. We can also see from the table that the statistical error, $\sigma(\mu_N^*)$, is dominated by the low order terms. Finally, it is clear that the contributions from the errors of the γ^* and A^* terms are much smaller than the statistical error for large N.

One additional point is that for the largest values of N, $N^{-\Delta_1-1}$ is of the order of 10^{-11}. In principle, this term could have a large constant and result in a large and unknown systematic

Table 2. Estimates of μ_N^* with statistical error $\sigma(\mu_N^*)$, and contributions to the systematic error.

N	μ_N^*	$\sigma(\mu_N^*)$	$\mu\sigma_\gamma \log N/N$	$\mu\sigma_A/(AN)$	$N^{-\Delta_1-1}$
73	4.683 732 537 07	1.70×10^{-8}	2.79×10^{-6}	1.07×10^{-4}	1.60×10^{-3}
147	4.683 926 584 87	2.13×10^{-8}	1.60×10^{-6}	5.28×10^{-5}	5.61×10^{-4}
295	4.684 000 343 15	2.40×10^{-8}	9.06×10^{-7}	2.62×10^{-5}	1.97×10^{-4}
591	4.684 026 832 89	2.53×10^{-8}	5.07×10^{-7}	1.31×10^{-5}	6.96×10^{-5}
1183	4.684 035 894 77	2.60×10^{-8}	2.80×10^{-7}	6.52×10^{-6}	2.46×10^{-5}
2367	4.684 038 837 75	2.64×10^{-8}	1.54×10^{-7}	3.26×10^{-6}	8.68×10^{-6}
4735	4.684 039 716 55	2.68×10^{-8}	8.37×10^{-8}	1.63×10^{-6}	3.07×10^{-6}
9471	4.684 039 940 72	2.70×10^{-8}	4.53×10^{-8}	8.14×10^{-7}	1.08×10^{-6}
18 943	4.684 039 975 88	2.71×10^{-8}	2.44×10^{-8}	4.07×10^{-7}	3.84×10^{-7}
37 887	4.684 039 965 93	2.71×10^{-8}	1.30×10^{-8}	2.04×10^{-7}	1.36×10^{-7}
75 775	4.684 039 954 43	2.71×10^{-8}	6.95×10^{-9}	1.02×10^{-7}	4.79×10^{-8}
151 551	4.684 039 943 95	2.71×10^{-8}	3.69×10^{-9}	5.09×10^{-8}	1.69×10^{-8}
303 103	4.684 039 937 49	2.72×10^{-8}	1.95×10^{-9}	2.54×10^{-8}	5.99×10^{-9}
606 207	4.684 039 934 06	2.72×10^{-8}	1.03×10^{-9}	1.27×10^{-8}	2.12×10^{-9}
1212 415	4.684 039 932 35	2.72×10^{-8}	5.41×10^{-10}	6.36×10^{-9}	7.49×10^{-10}
2424 831	4.684 039 931 45	2.72×10^{-8}	2.84×10^{-10}	3.18×10^{-9}	2.65×10^{-10}
4849 663	4.684 039 930 96	2.72×10^{-8}	1.49×10^{-10}	1.59×10^{-9}	9.36×10^{-11}
9699 327	4.684 039 930 69	2.72×10^{-8}	7.77×10^{-11}	7.95×10^{-10}	3.31×10^{-11}
19 398 655	4.684 039 930 58	2.72×10^{-8}	4.05×10^{-11}	3.97×10^{-10}	1.17×10^{-11}
38 797 311	4.684 039 930 52	2.72×10^{-8}	2.11×10^{-11}	1.99×10^{-10}	4.14×10^{-12}

error. In practice, because of the smooth convergence of our estimates we know that the constant cannot be large, and hence contributions from this term to μ_N^* are negligible for large N.

We thus conclude that the estimate $\mu^*_{38\,797\,311}$ has negligible systematic error, and hence adopt this as our best estimate for μ. Our final estimate is $\mu = 4.684\,039\,931(27)$.

Note, we *could* have avoided the use of previous estimates of γ and A, had the calculation of $\widetilde{B}_N$ been extended to larger N. This was not done because for N of the order of 100 million or so, both memory management and initialization time become significant but not insurmountable issues for the simulation of SAW using the SAW-tree implementation [16].

4. Discussion

As noted in the introduction, for the calculation of μ the approach which is most competitive with the algorithm presented in this paper is PERM [10], where the estimate $\mu = 4.684\,0386(11)$ was obtained. Our error bar is approximately 40 times smaller, which is clearly a significant improvement upon the previous state of the art. Other approaches to the calculation of μ worth noting are the method of atmospheres [19], and the Berretti–Sokal algorithm [20].

The use of concatenation in concert with the pivot algorithm was mentioned by Madras and Sokal [13], and pursued by Caracciolo *et al* [21] who made an accurate estimate of the critical exponent γ via the join-and-cut algorithm.

We note in passing that the method of atmospheres could be combined with the pivot algorithm and scale-free moves to obtain an accurate estimate for μ. We will not go into any depth, but the method of atmospheres corresponds to estimating

$$\frac{c_{N+K}}{c_N c_K} \sim \frac{A\mu^{N+K}}{A\mu^N c_K} = \frac{\mu^K}{c_K}, \tag{4.1}$$

for small, fixed K, and in the limit $N \to \infty$. From this expression one can then estimate μ once corrections to scaling have been taken into account. Despite being more accurate than previous methods, it is an order of magnitude less accurate than the method described here. This is because the mean CPU time per pivot attempt is $O(\log N)$ for the SAW-tree implementation. For the atmospheric sampling method, the dominant error comes from sampling walks in the large N limit, while for the method described in this paper the dominant error term comes from sampling short walks (in our case, with $N = 36$).

On the topic of approximation enumeration of SAW beyond the limit of exact enumeration, there have been a number of papers in recent years. Approaches include incomplete enumeration [22], flatPERM and flatGARM [23], stochastic enumeration [24], and the multicanonical Monte Carlo method [25]. The relative advantage of our approach is significant for small N, e.g. Shirai and Kikuchi [25] obtained $c_{256} = 6.2(4) \times 10^{108}$ for the square lattice, while for comparison we found $c_{295} = 2.014\,793(3) \times 10^{198}$ on the simple cubic lattice. For larger N, the relative advantage of our method increases, since to generate a SAW using an incremental growth method takes CPU time at least $O(N)$. This factor of N becomes prohibitively large when N is of the order of millions.

It is not clear to us if our approach could be adapted to other approximate enumeration problems, or to estimations of the free energy for other models in statistical mechanics. The general principles of 'divide-and-conquer' and the use of global moves in the canonical ensemble may be of wider use, or it may be that SAW is a particularly favorable model.

We consider figure 3 to be strong evidence in favor of the correctness of conjecture 1. We therefore expect that the use of scale-free moves for the simulation of polymers will prove useful in other contexts where there are additional length scales. For example, in the cases of star polymers or confined polymers. We will explore this idea further in a future paper where we will also derive an estimate of the critical exponent γ [18].

In future, our implementation of the SAW-tree could be optimized for the non-uniform selection of pivot sites according to our scale-free prescription. In particular, there is no reason a pivot being performed near the end of a walk should take mean CPU time $O(\log N)$. It is possible to arrange the binary tree data structure so that this operation would take time $O(1)$. One natural way of doing this would be to use a splay tree [26], which would dynamically adjust to form an optimal tree structure for any choice of pivot site sampling distribution.

We could also obtain a constant factor improvement, if it were possible to efficiently forbid configurations with immediate returns at the concatenation joint.

Finally, it is certainly possible to apply this approach to other lattices. Unfortunately, in the case of the square lattice the finite lattice method enumerations of polygons provide estimates for μ [9] which are approximately two orders of magnitude more accurate than our method. However, for three-dimensional lattices such as the body centered cubic lattice and the face centered cubic lattice, our method will allow for much more accurate calculations of μ than are currently available.

5. Conclusion

We have applied the pivot algorithm to calculate the connective constant for self-avoiding walks on the simple cubic lattice, obtaining $\mu = 4.684\,039\,931(27)$. Our approach may also be used to derive extremely accurate estimates for the number of self-avoiding walks. The power of our approach derives from the application of an efficient global move (the pivot algorithm), use of an observable which is calculated through a divide-and-conquer approach, and from the application of scale-free moves. We hope that these key ideas may prove useful in other contexts.

Acknowledgment

Financial support from the ARC Centre of Excellence for Mathematics and Statistics of Complex Systems is gratefully acknowledged.

References

[1] Madras N and Slade G 1993 *The Self-Avoiding Walk* (Boston, MA: Birkhaüser)

[2] Bauerschmidt R, Duminil-Copin H, Goodman J and Slade G 2010 Lectures on self-avoiding walks *Probability and Statistical Physics in Two and More Dimensions (Clay Mathematics Proceedings* vol 15*)* ed D Ellwood *et al* (Providence, RI: American Mathematical Society) pp 395–467

[3] Guttmann A J (ed) 2009 *Polygons, Polyominoes and Polycubes* vol 775 (Berlin: Springer)

[4] Caracciolo S, Guttmann A J, Jensen I, Pelissetto A, Rogers A N and Sokal A D 2005 Correction-to-scaling exponents for two-dimensional self-avoiding walks *J. Stat. Phys.* **120** 1037–100

[5] Clisby N, Liang R and Slade G 2007 Self-avoiding walk enumeration via the lace expansion *J. Phys. A: Math. Theor.* **40** 10973–1017

[6] de Neef T 1975 *PhD Thesis* Eindhoven University of Technology

[7] de Neef T and Enting I G 1977 Series expansions from the finite lattice method *J. Phys. A: Math. Gen.* **10** 801–5

[8] Enting I G 1980 Generating functions for enumerating self-avoiding rings on the square lattice *J. Phys. A: Math. Gen.* **13** 3713–22

[9] Clisby N and Jensen I 2012 A new transfer-matrix algorithm for exact enumerations: self-avoiding polygons on the square lattice *J. Phys. A: Math. Theor.* **45** 115202

[10] Grassberger P 2005 Simulations of grafted polymers in a good solvent *J Phys. A: Math. Gen.* **38** 323–31

[11] Schram R D, Barkema G T and Bisseling R H 2011 Exact enumeration of self-avoiding walks *J. Stat. Mech.* **2011** P06019

[12] Lal M 1969 'Monte Carlo' computer simulation of chain molecules: I *Mol. Phys.* **17** 57–64

[13] Madras N and Sokal A D 1988 The pivot algorithm: a highly efficient Monte Carlo method for the self-avoiding walk *J. Stat. Phys.* **50** 109–86

[14] Kennedy T 2002 A faster implementation of the pivot algorithm for self-avoiding walks *J. Stat. Phys.* **106** 407–29

[15] Clisby N 2010 Accurate estimate of the critical exponent ν for self-avoiding walks via a fast implementation of the pivot algorithm *Phys. Rev. Lett.* **104** 055702

[16] Clisby N 2010 Efficient implementation of the pivot algorithm for self-avoiding walks *J. Stat. Phys.* **140** 349–92

[17] Owczarek A L and Prellberg T 2008 Scaling of the atmosphere of self-avoiding walks *J. Phys. A: Math. Theor.* **41** 375004

[18] Clisby N 2013 Scale-free Monte Carlo method for calculating the critical exponent γ of self-avoiding walks (in preparation)

[19] Rechnitzer A and Janse van Rensburg E J 2002 Canonical Monte Carlo determination of the connective constant of self-avoiding walks *J. Phys. A: Math. Gen.* **35** L605–12

[20] Berretti A and Sokal A D 1985 New Monte Carlo method for the self-avoiding walk *J. Stat. Phys.* **40** 483–531

[21] Caracciolo S, Pelissetto A and Sokal A D 1992 Join-and-cut algorithm for self-avoiding walks with variable length and free endpoints *J. Stat. Phys.* **67** 65–111

[22] Sumedha and Dhar D 2005 Efficiency of the incomplete enumeration algorithm for Monte-Carlo simulation of linear and branched polymers *J. Stat. Phys.* **120** 71–100

[23] Janse van Rensburg E J 2010 Approximate enumeration of self-avoiding walks *Algorithmic Probab. Comb.* **520** 127–51

[24] Rubinstein R 2012 Stochastic enumeration method for counting NP-hard problems *Methodol. Comput. Appl. Probab.* **15** 249–91

[25] Shirai N C and Kikuchi M 2012 Multicanonical simulation of the Domb–Joyce model and the Go model: new enumeration methods for self-avoiding walks *CCP2012: Proc. Conf. on Computational Physics* at press (arXiv:1212.2181)

[26] Sleator D D and Tarjan R E 1985 Self-adjusting binary search trees *J. ACM* **32** 652–86