Testable Algorithms for Self-Avoiding Walks

DANA RANDALL[†] ALISTAIR SINCLAIR[‡]

Abstract

We present a polynomial time Monte Carlo algorithm for almost uniformly generating and approximately counting self-avoiding walks in rectangular lattices $\mathbb{Z}^d$. These are classical problems that arise, for example, in the study of long polymer chains. While there are a number of Monte Carlo algorithms used to solve these problems in practice, these are heuristic and their correctness relies on unproven conjectures. In contrast, our algorithm depends on a single, widely-believed conjecture that is weaker than preceding assumptions, and, more importantly, is one which the algorithm itself can test. Thus our algorithm is *reliable*, in the sense that it either outputs answers that are guaranteed, with high probability, to be correct, or finds a counter-example to the conjecture.

1 Summary

1.1 Background

A *self-avoiding walk* in a graph is a walk which starts at a fixed origin and passes through each vertex at most once. This paper is concerned with self-avoiding walks in lattices, in particular the d-dimensional rectangular lattice $\mathbb{Z}^d$ with origin $\mathbf{0}$.

Self-avoiding walks in $\mathbb{Z}^d$ have been studied by mathematicians and natural scientists for many years and are the subject of an extensive literature; for a state-of-the-art survey, see the recent book of Madras and Slade [16]. (See also the book by Lawler [14] for related topics.) One of the most important applications is as a model for the spatial arrangement of linear polymer molecules in chemical physics. Here the walk represents a molecule composed of many (perhaps 10^5 or more) monomers linked in a chain, and the self-avoidance constraint reflects the fact that no two monomers may occupy the same position in space.

The *length* $|w|$ of a self-avoiding walk w is the number of edges in w. For any fixed dimension d, let $\mathcal{S}_n$ denote the set of self-avoiding walks of length n in $\mathbb{Z}^d$, and let $c_n = |\mathcal{S}_n|$ be the number of walks of length n. The two most fundamental computational problems concerning self-avoiding walks are:

(i) count the number of walks of length n: i.e., compute c_n for any given n;

(ii) determine the characteristics of a "typical" walk of length n: for example, compute the *mean-square displacement*, which is the expected squared distance of the free end of the walk from the origin under the uniform probability distribution over walks of length n.

Despite much research in this area, and many heuristic arguments and empirical studies, almost nothing is known in rigorous terms about the above problems for the most interesting cases of low-dimensional lattices with $2 \leq d \leq 4$. In higher dimensions rather more is known, essentially because the self-avoidance constraint becomes less significant and the behavior resembles that of simple (non-self-avoiding) walks, which are well understood. Thus although the algorithmic results we present in this paper will be stated for arbitrary dimensions d, they are of greatest interest in the case of low-dimensional lattices with $2 \leq d \leq 4$.

One key fact that holds in all dimensions was discovered in 1954 by Hammersley and Morton [8]; they

[†]Department of Computer Science, University of California at Berkeley. Supported in part by an AT&T PhD Fellowship and by NSF grant CCR88-13632. E-mail: `randall@cs.berkeley.edu`.

[‡]University of Edinburgh and International Computer Science Institute, Berkeley. Supported in part by grant GR/F 90363 of the UK Science and Engineering Research Council, and by Esprit Working Group "RAND." E-mail: `sinclair@icsi.berkeley.edu`.

observed that $\lim_{n\to\infty} c_n^{1/n} = \mu$ exists, and that $\mu^n \leq c_n = \mu^n f(n)$, where $\lim_{n\to\infty} f(n)^{1/n} = 1$. This is a straightforward consequence of the obvious fact that the sequence $\ell_n = \log c_n$ is *subadditive*, i.e., $\ell_{n+m} \leq \ell_n + \ell_m$ for all n, m. Hammersley and Welsh [9] later showed that $f(n) = \mathrm{O}(a^{n^{1/2}})$ for some constant a. It is a celebrated and long-standing conjecture that $f(n)$ is in fact polynomially bounded, and more precisely that

$$c_n = \mu^n \widetilde{f}(n)(1 + \mathrm{o}(1))$$

where

$$\widetilde{f}(n) = \begin{cases} An^{\gamma-1}, & d = 2, 3; \\ A(\log n)^{1/4}, & d = 4; \\ A, & d \geq 5. \end{cases} \qquad \text{(C1)}$$

Here μ, A and γ are all dimension-dependent constants. Note that the dominant behavior of c_n is the exponential function μ^n; comparing this with the case of simple walks, whose number is precisely $(2d)^n$, we see that the effect of the self-avoidance constraint is to reduce the effective number of choices the walk has at each step from $2d$ to μ. The dimension-dependent number μ is known as the *connective constant.* This crude behavior is modified by the correction term $f(n)$ of the form conjectured in C1. Here γ is a so-called *critical exponent.* (Note, however, that γ, unlike μ, is not even known to exist.)

Although unproven, conjecture C1 is supported by extensive (though non-rigorous) empirical studies and ingenious heuristic arguments, which have also been employed to obtain numerical estimates for the constants μ and γ. Elementary considerations show that $\mu \in (d, 2d-1)$. For $d = 2$, it has actually been proven that $\mu \in (2.62, 2.70)$ [1, 4]. (See also [10] for similar bounds in higher dimensions.) However, these rigorous bounds are much weaker than the non-rigorous estimates obtained by empirical methods, which are typically quoted to four decimal places. There are even precise conjectured values for the critical exponent γ in two and three dimensions (despite the fact that γ is not known to exist): for $d = 2$, γ is believed to be $\frac{43}{32}$, and for $d = 3$ it is believed to be approximately 1.16. (See [16] for a detailed summary of numerical estimates.)

Much effort has been invested in obtaining statistical estimates of the above quantities using Monte Carlo simulations. However, the error bars on these estimates are only justified heuristically. In this paper, we attempt to put such experiments on a firmer footing. We present Monte Carlo algorithms for approximating the number of self-avoiding walks of a given length for a given dimension d, and for generating self-avoiding walks of a given length almost uniformly at random. The running time of our algorithms is polynomial in the walk length n and grows only slowly with parameters controlling the accuracy and confidence levels of the estimates. These are the first polynomial time algorithms where the statistical errors are rigorously controlled. Our algorithms are based on modifications and extensions of a Monte Carlo approach studied originally by Berretti and Sokal [2]. In the next subsection we sketch this approach and point out its limitations. Then, in section 1.3, we summarize our algorithms and explain how they overcome these problems.

1.2 Monte Carlo methods

Monte Carlo simulations have proved to be a powerful tool for developing approximation algorithms for a range of combinatorial problems. Briefly, the idea is as follows. Let $\mathcal{S}$ be a large but finite set of combinatorial structures. It is well known that much information about $\mathcal{S}$ can be gained by sampling elements of $\mathcal{S}$ from an appropriate probability distribution π. This sampling can be performed by simulating a *Markov chain* whose state space includes $\mathcal{S}$ and whose conditional stationary distribution over $\mathcal{S}$ is π: to get a sample from a distribution very close to π, one simply simulates the chain for sufficiently many steps that it is close to stationarity, and outputs the final state if it belongs to $\mathcal{S}$. In order for this method to be effective, the stationary distribution must be reasonably well concentrated on $\mathcal{S}$ (so that one gets a valid sample reasonably often), and the Markov chain must converge rapidly to its stationary distribution (so that the number of simulation steps required is not too large).

In the case of self-avoiding walks, we are interested in sampling from the uniform distribution over the set $\mathcal{S}_n$ of walks of length n. A natural Markov chain to use here has as its state space the set of all self-avoiding walks (of all lengths): if the chain is currently at a walk w, it extends the walk in an allowable direction with some probability, while with some other probability it deletes the last edge and "backtracks" to a shorter walk. Note that the naïve approach of simply growing the walk one edge at a time breaks down because of the self-avoidance constraint: the number of possible extensions of a given length can vary hugely for different walks due to the possibility of walks "getting stuck." This is why we require the more sophisticated dynamic scheme provided by the Markov chain.

The above type of Markov chain was considered by Berretti and Sokal [2], who used a single parameter $\beta \leq 1$ to control the relative probabilities of extending or contracting the walk by one edge. Given a walk of length i, one of the $2d$ lattice edges incident to the free endpoint of the walk is chosen with equal probability. If the edge extends the walk so as to be self-avoiding, then it is added with probability β; if the edge is the last edge of the walk, then it is removed; otherwise, nothing is done.[†] Assuming conjecture C1, Berretti and Sokal argue that, for any given value of n, taking β sufficiently close to (but smaller than) μ^{-1}, where μ is the connective constant, ensures that the stationary distribution assigns reasonably high weight (i.e., $1/q(n)$ for some polynomial q) to $\mathcal{S}_n$. Furthermore, again assuming conjecture C1, Sokal and Thomas [21] argue that with such values of β the Markov chain is *rapidly mixing*, i.e., it gets very close to stationarity after a number of steps that is only polynomial in n (see also [15]). In order to appreciate the role of β here, consider a truncated version of this Markov chain in which the length of a walk is never allowed to exceed n, so that the stationary distribution is always well defined; if β is too much smaller than μ^{-1} then we will only generate short walks, while if β is too much larger then the Markov chain will not backtrack often enough and consequently will take a long time to reach stationarity. Thus β must be very carefully chosen. Berretti and Sokal perform their experiments by "fine-tuning" β and observing the Markov chain until the observations suggest that β is sufficiently close to μ^{-1}.

Berretti and Sokal's algorithm suffers from two drawbacks. First, one must assume conjecture C1 (for appropriate values of the constants μ, γ and A) in order to bound the time required before the Markov chain reaches stationarity. As long as conjecture C1 remains open for any choices of these constants, there is no guarantee that the algorithm produces reliable answers in polynomial time. Second, in order to implement the algorithm it is necessary to have a good estimate of μ already, since β needs to be taken a little smaller than μ^{-1}. This leads to circularity, since determining μ is one of the principal goals of the algorithm. While many similar Monte Carlo algorithms have been used to study self-avoiding walks (see Chapter 9 of [16] for a summary), all of these suffer from a similar lack of rigorous justification, and thus offer no guarantee that their results are reliable.

1.3 Our results

In this paper we develop a Monte Carlo algorithm for self-avoiding walks by modifying the Markov chain used by Berretti and Sokal so as to overcome the difficulties discussed in the last subsection. We make two elementary but important innovations. First, we allow the parameter β to vary at each level of the Markov chain (i.e., we let β depend on the length of the walks), and we *calculate* an appropriate value of β at each level based on observations of the Markov chain at earlier levels. Thus we require no prior knowledge of β. Second, we show that, while the efficiency of our Markov chain simulation is still based on an assumption (conjecture C2, defined below), this is weaker than conjecture C1 and, more importantly, is tested in advance by the algorithm in the region in which it is being assumed. Thus when we run our algorithm, *either* we will gather strong evidence (in the form of a counter-example) that the conjecture is false, *or* we will know that we can trust our simulations. This notion of *self-testing*, which either gives us confidence in our results or warns us that they may be erroneous, has been previously explored in the context of program checking (see, e.g., [3]).

The conjecture we require is the following, for a given dimension d:

$$\text{for some fixed polynomial } g, \qquad c_n c_m \le g(n+m)\, c_{n+m}, \quad \forall n,m. \tag{C2}$$

This conjecture says that, if we choose random self-avoiding walks of lengths n and m, then with non-negligible probability we can glue the walks together to produce a self-avoiding walk of length $n+m$. Conjecture C2 is no more restrictive than conjecture C1, on which previous Monte Carlo methods, including that of Berretti and Sokal, rely. To see this, note that $c_n \sim A\mu^n n^{\gamma-1}$ implies $\frac{c_n c_m}{c_{n+m}} \sim A(\frac{nm}{n+m})^{\gamma-1} \le A\left(\frac{n+m}{4}\right)^{\gamma-1}$. Thus conjecture C2 is also widely believed to hold. Moreover, for any given dimension there is a precise conjectured value for the polynomial g: as the above calculation shows, it is essentially just the function $\tilde{f}$ from conjecture C1, with appropriate values for the constants μ, γ and A.

The behavior of our algorithm may now be stated more precisely as follows. Fix a dimension d and a polynomial g, and suppose first that conjecture C2 holds. Then, on inputs $\epsilon, \delta \in (0,1)$, the algorithm outputs a sequence of numbers $\tilde{c}_1, \tilde{c}_2, \tilde{c}_3, \ldots$, such that, for each n, the time to output $\tilde{c}_n$ is a polynomial function of n, ϵ^{-1} and $\log\delta^{-1}$ and, with probability at least $(1-\delta)$, $\tilde{c}_n$ approximates c_n within ratio $(1+\epsilon)$. An algorithm with this behavior is an example of a *fully-polynomial randomized approximation scheme* for the number of self-avoiding walks. If, on the other hand, the conjecture happens to fail for some value $n = n_0$, then with high probability an error will be reported and we will know that the algorithm is reliable in the region previously explored (i.e., for $n < n_0$), but may be unreliable for larger values of n. Moreover, in this case the algorithm will actually have discovered a counter-example to the conjecture for the polynomial g under consideration; since precise conjectured values for g exist, this in itself would be of substantial interest in the theory of self-avoiding walks. The details of the self-tester are described explicitly in section 3. Note that, in the presence of the self-tester, the answers output by our algorithm are always correct (with high probability), and the algorithm is guaranteed always to run in polynomial time.

[†]Actually, these transition probabilities are a slightly simplified version of those used in [2], but this difference is inessential to the behavior of the chain.

The algorithm is in fact more flexible and can be used in addition to solve problem (ii) of section 1.1 by generating random self-avoiding walks of any specified length in the region where conjecture C2 holds: once the algorithm has output $\widetilde{c}_n$, it provides a method for generating, in time polynomial in n and $\log\delta^{-1}$, a random self-avoiding walk of length n from a distribution whose variation distance from the uniform distribution is at most δ. Such an algorithm is called a *fully-polynomial almost uniform generator* for self-avoiding walks, and can be used in the obvious fashion to obtain good statistical estimates in polynomial time of quantities such as the mean-square displacement.

In section 2 we present approximation algorithms which work assuming conjecture C2. In section 3 we show how to make the algorithms robust by adding a self-tester to verify the conjecture.

2 The algorithms

First let us make more precise the properties we want our algorithms to have. The following definitions of approximation algorithms for counting and uniform generation have been used in [13, 12, 18] and elsewhere.

Definition. **(i)** A *randomized approximation scheme* for the number of self-avoiding walks in some fixed dimension d is a probabilistic algorithm which, on input n and $\epsilon, \delta \in (0,1)$, outputs a number $\tilde{c}$ such that $\Pr\{c_n(1+\epsilon)^{-1} \le \tilde{c} \le c_n(1+\epsilon)\} \ge 1-\delta$. The approximation scheme is *fully-polynomial* if it is guaranteed to run in time polynomial in n, ϵ^{-1} and $\log\delta^{-1}$.
(ii) An *almost uniform generator* for self-avoiding walks is a probabilistic algorithm which, on input n and $\epsilon \in (0,1)$, outputs a self-avoiding walk of length n with probability at least $1/q(n)$ for a fixed polynomial q, such that the conditional probability distribution over walks of length n has variation distance at most ϵ from the uniform distribution. The generator is *fully-polynomial* if it runs in time polynomial in n and $\log\epsilon^{-1}$. $\square$

The following quantity associated with self-avoiding walks will play an important role in what follows. For a fixed dimension d and each n, define

$$\alpha_n = \min_{\substack{j,k \\ j+k\le n}} \frac{c_{j+k}}{c_j c_k}.$$

Note that conjecture C2 says precisely that $\alpha_n \ge g(n)^{-1}$ for a polynomial g (which depends on d).

The quantity α_n may be interpreted as follows. For fixed j and k, $\frac{c_{j+k}}{c_j c_k}$ represents the probability that a random self-avoiding walk of length j and a random self-avoiding walk of length k can be "glued" together to form a self-avoiding walk of length $j+k$. To be more precise, for self-avoiding walks w_1 and w_2, define the *concatenation* $w_1 \circ w_2$ to be the walk formed by translating w_2 so that its origin coincides with the free endpoint of w_1 and appending the translated copy of w_2 to w_1. Note that $w_1 \circ w_2$ need not be self-avoiding. If w_1 and w_2 are selected uniformly at random from $\mathcal{S}_j$ and $\mathcal{S}_k$ respectively, then the above quotient represents the probability that $w_1 \circ w_2$ *is* self-avoiding.

Theorem 1 *For any fixed dimension d, there exists a randomized approximation scheme for self-avoiding walks that runs in time polynomial in $n, \epsilon^{-1}, \log\delta^{-1}$ and α_n^{-1}, and an almost uniform generator that runs in time polynomial in $n, \log\epsilon^{-1}$and α_n^{-1}.*

It is interesting to observe that this result, combined with the asymptotic bound on c_n of Hammersley and Welsh [9] quoted in section 1.1, immediately gives us approximation algorithms for self-avoiding walks whose running time is sub-exponential. Specifically, the bound of [9] implies that $\alpha_n^{-1} = \mathrm{O}(a^{n^{1/2}})$ for some constant a, and closer inspection of the running time of the algorithms of theorem 1 (see section 2.3) reveals that the dependence on α_n^{-1} is linear. Thus we get a randomized approximation scheme and an almost uniform generator whose running times grow with n only as $\exp(\mathrm{O}(n^{1/2}))$.

If we assume conjecture C2, however, we get something much stronger. Since conjecture C2 asserts that $\alpha_n \ge g(n)^{-1}$ for a polynomial g, we immediately deduce the following.

Corollary 2 *Assuming conjecture C2, there exists a fully-polynomial randomized approximation scheme and a fully-polynomial almost uniform generator for self-avoiding walks in any fixed dimension d.* $\square$

Our algorithms are based on randomly sampling walks of length n using Monte Carlo simulation of a series of successively larger Markov chains $M_1, ..., M_n$. In section 2.1 we define the nth Markov chain M_n, and in section 2.2 we derive a bound on its rate of convergence to stationarity. With this machinery in place, in section 2.3 we assemble the Markov chains into a single scheme that provides algorithms satisfying the requirements of theorem 1.

2.1 The Markov chain

As indicated in section 1.2, we consider a Markov chain that explores the space of self-avoiding walks by letting a walk expand and contract randomly over time, under the influence of a weighting parameter β. Rather than working with a single Markov chain and a global value of the parameter β, we incrementally construct Markov chains $M_1, M_2, \ldots$, the nth of which, M_n, has as its state space the set $\mathcal{X}_n = \bigcup_{i=0}^{n} \mathcal{S}_i$ of all self-avoiding walks of length at most n. The transition probabilities in M_n depend on parameters $\beta_1, ..., \beta_n \in (0,1)$, discussed below.

Transitions in the Markov chain M_n are defined as follows. In state $w \in \mathcal{X}_n$, a self-avoiding walk of length $i \leq n$, choose one of the $2d$ edges incident to the free endpoint of w uniformly at random. If the chosen edge coincides with the last step of w, remove this last edge from w. If the chosen edge extends w to a walk which is self-avoiding and has length at most n, add the edge to w with probability β_{i+1}. Otherwise, leave w unchanged.

More precisely, define the partial order $\prec$ on the set of all self-avoiding walks by $w \prec w'$ if and only if $|w| < |w'|$ and the first $|w|$ steps of w' coincide with w. Also, define $w \prec_1 w'$ if $w \prec w'$ and $|w'| = |w| + 1$ (i.e., if w' extends w by one step). Then the transition probabilities P_n of the Markov chain M_n are defined by

$$P_n(w, w') = \begin{cases} \beta_{|w'|}/2d, & \text{if } w \prec_1 w'; \\ 1/2d, & \text{if } w' \prec_1 w; \\ r(w), & \text{if } w = w'; \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where $r(w)$ is chosen so as to make the probabilities sum to 1, and w, w' are in the state space $\mathcal{X}_n$ (i.e., $|w|, |w'| \leq n$).

Note that we may view M_n as a weighted random walk on the tree defined by the partial order $\prec$. This tree has the trivial walk of length 0 at the root, and the children of walk w are walks w' with $w \prec_1 w'$. Thus the tree has $n+1$ levels, the ith of which contains all walks of length $i-1$. The transition probability from any state to its parent is $1/2d$, and from a state at level i to each of its children is $\beta_{i+1}/2d$. In the case that $\beta_1 = ... = \beta_n \approx \mu^{-1}$ this is just the Markov chain used by Berretti and Sokal [2], but truncated at level n.

It is evident that the Markov chain M_n is irreducible (all states communicate) and aperiodic. This implies that it is *ergodic*, i.e., it converges asymptotically to a well-defined equilibrium or *stationary distribution* π_n over $\mathcal{X}_n$. Thus, if $P^t(x, w)$ denotes the probability that the chain is in state w after t steps starting in some specified initial state x, then $P^t(x, w) \to \pi_n(w)$ as $t \to \infty$, for every $w \in \mathcal{X}_n$. It is straightforward to show the following:

Lemma 3 *The stationary distribution π_n of the Markov chain M_n is given by*

$$\pi_n(w) = \frac{1}{Z_n} \prod_{i=1}^{|w|} \beta_i, \text{ for } w \in \mathcal{X}_n,$$

where Z_n is a normalizing factor.

Proof. It suffices to show that the chain is *reversible* with respect to the distribution π_n, i.e., that it satisfies the *detailed balance* condition

$$\pi_n(w) P_n(w, w') = \pi_n(w') P_n(w', w) \quad \forall w, w' \in \mathcal{X}_n.$$

This is readily verified from the definition of P_n given in (1). □

Note that the stationary distribution is always uniform over all walks of a given length, for any choice of values of the parameters β_i. However, by choosing the β_i carefully we can achieve a distribution over *lengths* which assigns sufficiently high weight to $\mathcal{S}_n$. Ideally, the value we want for β_i is the ratio c_{i-1}/c_i. (The fact that this ratio is always strictly less than 1 was proven surprisingly recently by O'Brien [17].) Of course, this is unrealistic since we do not know the quantities c_{i-1} and c_i—indeed, these are precisely what we are trying to compute—but we will see in section 2.3 how to determine good approximations to these ideal values before they are needed. For the moment, we consider the ideal behavior of the Markov chain assuming that each β_i is equal to c_{i-1}/c_i.

Under this assumption, lemma 3 says that the stationary probability of any walk $w \in \mathcal{X}_n$ is

$$\pi_n(w) = \frac{1}{Z_n} \prod_{i=1}^{|w|} \frac{c_{i-1}}{c_i} = \frac{1}{Z_n c_{|w|}}. \tag{2}$$

Thus the stationary distribution is uniform over lengths, and the probability of being at a walk of length i is $1/Z_n = 1/(n+1)$ for each i. This means that the Markov chain M_n has the first of the two properties identified in section 1.3 that are required for the Monte Carlo approach to be effective: the stationary distribution is reasonably well concentrated on $\mathcal{S}_n$, and uniform over $\mathcal{S}_n$. We may therefore, at least in principle, generate random self-avoiding walks of length n by simulating M_n until it has reached equilibruim, starting with, say, the empty walk, and outputting the final state if it has length n. The second property required of the Markov chain is that the number of simulation steps should be small. This is the key component of the running time of our algorithms and is quantified in the next subsection.

2.2 The mixing time

The question of how many simulation steps are required to produce a sample from a distribution that is very close to π_n is precisely that of how long it takes for the Markov chain to get close to equilibrium. This is often referred to as the *mixing time*. Note that, if the overall running time of our algorithm is to be polynomial in n, the Markov chain M_n should be *rapidly mixing*, in the sense that its mixing time is very small compared to the number of states (which grows exponentially with n).

In recent years several useful analytical tools have been devised for analyzing the mixing time of complex

Markov chains of this kind. In this paper we make use of the idea of "canonical paths", first developed in [11, 18]. Consider an ergodic, reversible Markov chain with state space X, transition probabilities P and stationary distribution π. We can view the chain as a weighted undirected graph G with vertex set X and an edge between each pair of vertices (states) x, y for which $P(x,y) > 0$. We give each oriented edge $e = (x,y)$ a "capacity" $Q(e) = Q(x,y) = \pi(x)P(x,y)$; note that, by detailed balance, $Q(x,y) = Q(y,x)$.

Now for each ordered pair of distinct vertices $x, y \in X$, we specify a *canonical path* γ_{xy} in the graph G from x to y. Then, for any such collection of paths $\Gamma = \{\gamma_{xy} : x, y \in X, x \neq y\}$, define

$$\rho(\Gamma) = \max_e \frac{1}{Q(e)} \sum_{\gamma_{xy} \ni e} \pi(x)\pi(y) \tag{3}$$

where the maximization is over oriented edges e. Thus ρ measures the maximum loading of any edge e by paths in Γ as a fraction of its capacity $Q(e)$, where the path from x to y carries "flow" $\pi(x)\pi(y)$. Note that the existence of a collection of paths Γ for which $\rho(\Gamma)$ is small implies an absence of bottlenecks in the graph, and hence suggests that the Markov chain should be rapidly mixing. This intuition can be formalized and a bound obtained on the mixing time in terms of the quantity $\rho = \min_\Gamma \rho(\Gamma)$, using a measure known as *conductance* [20]. However, we can get a slightly sharper bound in this case by following an idea of Diaconis and Stroock [5] and using the alternative measure $\overline{\rho} = \min_\Gamma \rho(\Gamma)\ell(\Gamma)$, where $\ell(\Gamma)$ is the maximum length of a path in Γ. The appropriate version of this bound can be found by combining Proposition 1 and Corollary 6 of [19] and is stated precisely in the theorem below.

As a measure of rate of convergence, let $P^t(x,\cdot)$ be the probability distribution of the Markov chain at time t, starting in state x, and for $\epsilon \in (0,1)$ define

$$\tau_x(\epsilon) = \min\{t : \|P^{t'}(x,\cdot) - \pi\| \leq \epsilon \quad \forall t' \geq t\}.$$

Here $\|\cdot\|$ denotes variation distance: for distributions ν_1, ν_2 over X, $\|\nu_1 - \nu_2\| = \frac{1}{2}\sum_{x \in X} |\nu_1(x) - \nu_2(x)| = \max_{A \subseteq X} |\nu_1(A) - \nu_2(A)|$.

Theorem 4 [19] *For an ergodic, reversible Markov chain with stationary distribution π, we have*

$$\tau_x(\epsilon) \leq K\,\overline{\rho}\Big(\log \pi(x)^{-1} + \log \epsilon^{-1}\Big),$$

where K is a universal constant. □

We now use theorem 4 to show that the mixing time of the Markov chain M_n can be bounded in terms of the quantity α_n defined at the beginning of section 2. Assuming conjecture C2, this will imply that the Markov chain is rapidly mixing. For simplicity we will work with the idealized version of M_n discussed at the end of section 2.1, in which each β_i is exactly equal to c_{i-1}/c_i. It should be clear that our analysis is not unduly sensitive to small perturbations in the values of the β_i.

Theorem 5 *For the Markov chain M_n, starting at the empty walk* $\mathbf{0}$, *we have*

$$\tau_{\mathbf{0}}(\epsilon) \leq Kdn^2\alpha_n^{-1}\Big(\log n + \log \epsilon^{-1}\Big)$$

for some constant K.

Proof. From (2) we have that $\pi_n(\mathbf{0}) = 1/(n+1)$. Also, since the graph corresponding to the Markov chain M_n is a tree, there is only one choice of (simple) paths between each pair of vertices; we will denote this collection of paths $\Gamma = \{\gamma_{xy}\}$. Since the depth of the tree is n, we have $\ell(\Gamma) = 2n$. Therefore, the result will follow from theorem 4 if we can show that $\rho(\Gamma) \leq K'dn\alpha_n^{-1}$ for some constant K'.

Now let e be any edge of the tree, and suppose the endpoints of e are a walk w of length k and a walk w' of length $k+1$. Let S be the subtree rooted at w', and $\overline{S} = \mathcal{X}_n - S$. Since e is a cut edge, it is clear that (3) becomes

$$\rho(\Gamma) = \max_e Q(e)^{-1}\pi_n(S)\pi_n(\overline{S}). \tag{4}$$

In what follows we will make essential use of the fact that the tree defining M_n is a *sub-Cayley* tree, so that the number of vertices at level l of any subtree is bounded above by the total number of vertices at level l of the whole tree. This is evident since any initial segment of a self-avoiding walk is also self-avoiding.

Now we have

$$Q(e) = \pi_n(w')P_n(w',w) = \frac{1}{2dZ_nc_{k+1}},$$

and

$$\begin{aligned}
\pi_n(S) &= \sum_{\tilde{w} \succeq w'} \pi_n(\tilde{w}) \\
&= \sum_{j=k+1}^{n} \frac{1}{Z_nc_j} |\{\tilde{w} \succeq w' : |\tilde{w}| = j\}| \\
&= \frac{1}{Z_nc_{k+1}} \sum_{j=k+1}^{n} \frac{c_{k+1}}{c_j} |\{\tilde{w} \succeq w' : |\tilde{w}| = j\}| \\
&\leq \frac{1}{Z_nc_{k+1}} \sum_{j=k+1}^{n} \frac{c_{k+1}\,c_{j-k-1}}{c_j} \\
&\leq \frac{n}{Z_nc_{k+1}\alpha_n},
\end{aligned}$$

where the first inequality follows from the sub-Cayley property of the tree. Putting this together, we see that $Q(e)^{-1}\pi_n(S)\pi_n(\overline{S}) \leq Q(e)^{-1}\pi_n(S) \leq 2dn\alpha_n^{-1}$. Since e was arbitrary, (4) now gives us the required upper bound on $\rho(\Gamma)$. □

Remark. A similar bound on the mixing time of the Berretti-Sokal Markov chain was obtained using ad-hoc methods by Sokal and Thomas [21]. Again the essential feature that makes the argument work is the sub-Cayley property of the tree underlying the chain. A rather weaker bound was obtained by Lawler and Sokal [15], using the discrete Cheeger inequality or conductance. This latter proof is very similar in spirit to the one above; essentially, the effect is to replace $\overline{\rho}$ by ρ^2 in the bound of theorem 4. □

2.3 The overall algorithm

In this subsection, we show how to assemble the sequence of Markov chains just described into a single algorithm that outputs a sequence of numbers $\{\widetilde{c}_n\}$, each of which is a good estimate of the corresponding c_n. The accuracy of the estimates is controlled by two parameters, ϵ and δ, exactly as in the definition of a randomized approximation scheme appearing at the beginning of section 2. We shall see that the algorithm provides both an approximation scheme and an almost uniform generator with the properties claimed in theorem 1.

The main new ingredient in the algorithm is a bootstrapping procedure for computing the parameters β_n governing the transition probabilities of the Markov chains. Recall that our analysis so far has assumed that $\beta_n = c_{n-1}/c_n$ for each n. However, these values are not available to us; in fact, calculating the quantities c_n is one of our main objectives. Instead, our overall algorithm computes *estimates* of these ideal values c_{n-1}/c_n for each n in turn, using the previous Markov chain M_{n-1}. This is consistent since the first time that β_n is required is in the Markov chain M_n.

The algorithm, spelled out in detail in figure 1, works in a sequence of stages corresponding to the iterations of the **for**-loop. We call stage n of the algorithm *good* if it runs to completion and computes a value β_n that approximates the value c_{n-1}/c_n within ratio $(1+\epsilon/4n^2)$, where ϵ, δ are the accuracy and confidence inputs.

Let us consider the operation of stage n in detail. To compute a good approximation β_n of the ratio c_{n-1}/c_n, we randomly sample walks of length $n-1$ using the Markov chain M_{n-1} and estimate the average number of one-step extensions of a walk: we can compute the number of one-step extensions of a given walk by explicitly checking each of the $2d-1$ possibilities. Note that, for a random walk, this is a bounded random variable taking values in $[0, 2d-1]$ with mean at least 1 (since $c_n > c_{n-1}$). The sample size is controlled by the parameter T_n. A simple generalization of the 0/1-estimator theorem (see [13]) to handle non-negative, bounded random variables shows that T_n need not be too large to obtain a good estimate with sufficiently high probability. Finally, since we are in fact sampling from the larger set $\mathcal{X}_{n-1}$, we need to generate a sample of size $2nT_n$ to ensure that, with high probability, we get at least T_n walks of length $n-1$; that this sample is large enough follows from the fact that, by (2), in the stationary distribution of the chain M_{n-1} walks of length $n-1$ have weight $1/n$.[†] In the algorithm of figure 1, we abort in the unlikely event that the sample does not contain enough walks of length $n-1$.

```
$\beta_1 = 1/2d$; $\widetilde{c}_1 = 2d$
**for** $n = 2, 3, 4, \ldots$ **do**
    using $M_{n-1}$, generate a sample of size $2nT_n$ from
        (close to) the distribution $\pi_{n-1}$ over $\mathcal{X}_{n-1}$
    let $Y$ be the set of walks in the sample with
        length $n-1$
    $\text{ext}(Y) = \sum_{w \in Y} |\{w' \in \mathcal{S}_n : w \prec_1 w'\}|$
    **if** $|Y| < T_n$ or $\text{ext}(Y) = 0$ **then abort**
    **else** set $\beta_n = |Y|/\text{ext}(Y)$
    **output** $\widetilde{c}_n = \widetilde{c}_{n-1}/\beta_n$
```

Figure 1: The algorithm

Summarizing the above discussion, we get:

Lemma 6 *In the algorithm of figure 1, assuming that stages $1, 2, \ldots, n-1$ are good, stage n is good with probability at least $(1-\delta/2n^2)$, provided the sample size T_n is at least $cn^4\epsilon^{-2}(\log n + \log \delta^{-1})$ for a suitable constant c (which depends on the dimension d).* □

The algorithm is designed so that, assuming all previous stages $1, 2, \ldots, n-1$ are good, stage n will be good with probability at least $(1-\delta/2n^2)$. The reason for this requirement is the following. If all stages $1, 2, \ldots, n$ are good, then the value $\widetilde{c}_n = \prod_{i=1}^n \beta_i^{-1}$ output by the algorithm at the end of stage n approximates $\prod_{i=1}^n (c_i/c_{i-1}) = c_n$ within ratio $\prod_{i=1}^n (1+\epsilon/4i^2) \leq 1+\epsilon$; moreover, this happens with probability at least $\prod_{i=1}^n (1-\delta/2i^2) \geq 1-\delta$. Thus we get a randomized approximation scheme for c_n, which was one of our principal goals. Moreover, by the end of stage n we have

[†]Actually the Markov chain we are simulating here is not precisely that analyzed in sections 2.1 and 2.2, since the parameters β_i will differ slightly from their ideal values. However, it should be clear from lemma 3 that the stationary distribution is always uniform within each level of the tree, and that, if all previous stages are good, then the distribution over levels of the tree differs from the uniform distribution by at most a constant factor.

computed values β_i for $1 \leq i \leq n$; thus we have constructed a Markov chain M_n which we can simulate to generate random self-avoiding walks of any length up to n. This was our second principal goal.

The running time of stage n of the algorithm is dominated by $2nT_n = 2cn^5\epsilon^{-2}(\log n + \log\delta^{-1})$ times the time required to produce a single sample from M_{n-1}. From our analysis in the previous subsection (theorem 5), this latter time is $\mathrm{O}\left(n^2\alpha_n^{-1}(\log n + \log\epsilon^{-1})\right)$ for any fixed dimension d.[‡] Run to the nth stage, the algorithm is therefore an approximation scheme satisfying the requirements of theorem 1. (Note that the exponent of n in the running time could be improved by a more refined statistical analysis.) By the same reasoning, simulating the Markov chain M_n for $\mathrm{O}\left(n^2\alpha_n^{-1}(\log n + \log\epsilon^{-1})\right)$ steps gives us the almost uniform generator claimed in theorem 1.

3 Making the algorithm self-testing

In this section we show how to place the algorithm of the previous section on a firmer theoretical basis by replacing our assumption of conjecture C2 by an algorithmic test of the conjecture. This is a particular instance of what we believe is a generally useful idea of using *self-testing* to make an algorithm whose correctness depends on a conjecture more robust.

Recall that we have reduced the problem of constructing polynomial time approximation algorithms for self-avoiding walks to that of verifying a single widely believed conjecture about the behavior of the walks. An important feature of this reduction is the structure of the conjecture. Conjecture C2 bounds the probability that one can glue together two random self-avoiding walks to produce a new self-avoiding walk; since the algorithm also lets us generate random self-avoiding walks, we can actually estimate this probability. For any specified polynomial g, this allows us to verify the conjecture (and therefore also the algorithm itself) for a new value of n by using the algorithm in the region in which it has already been tested. This is precisely the idea behind the self-tester which we introduce in this section.

We showed in the last section how to construct a sequence of Markov chains for uniformly generating and counting self-avoiding walks of successively longer lengths. The running time of these algorithms is polynomial in the walk length n and the unknown parameter α_n^{-1}; this quantity enters into the time bound because it governs the mixing time of the Markov chains (see theorem 5). We then appealed to conjecture C2 to argue that α_n^{-1} is itself polynomially bounded in n. The idea behind the self-tester is to obtain a good estimate for α_n^{-1} in advance, so that we know how long to simulate our Markov chains to ensure our samples are sufficiently close to the stationary distribution. This will give us a probabilistic *guarantee* that the algorithm is correct, independent of conjecture C2. Moreover, by examining the growth rate of successive values α_n^{-1}, we can simultaneously *test* conjecture C2. The algorithm will proceed successfully for as long as we never exceed some projected upper bound $g(n)$ on α_n^{-1}; however, should α_n^{-1} grow too quickly we will detect this fact and we will have found a counter-example to the conjecture.

```
$\tilde\alpha_n = \tilde\alpha_{n-1}$
**for** $i = 1, 2, \ldots, n-1$ **do**
    **repeat** $t$ **times**
        generate $u_i \in \mathcal{S}_i$ u.a.r.
        generate $v_i \in \mathcal{S}_{n-i}$ u.a.r.
        $X_i = \begin{cases} 1 & \text{if } u_i \circ v_i \in \mathcal{S}_n; \\ 0 & \text{otherwise} \end{cases}$
    $q_{n,i} = \sum_i X_i/t$
    $\tilde\alpha_n = \min\{\tilde\alpha_n, q_{n,i}/2\}$
**if** $\tilde\alpha_n^{-1} > 4g(n)$
**then output** "Warning: conjecture fails"
**else continue**
```

Figure 2: The self-tester

More precisely, after each stage n of the algorithm of the previous section, we use the procedure shown in figure 2 to compute a quantity $\tilde\alpha_n$ such that, with high probability, $\alpha_n/4 \leq \tilde\alpha_n \leq \alpha_n$. The conservative estimate $\tilde\alpha_n^{-1}$ is then used in place of α_n^{-1} in the next stage when computing the mixing time of the Markov chain M_n from theorem 5. We also test conjecture C2 by comparing our estimates $\tilde\alpha_n^{-1}$ at each stage with (a constant multiple of) the projected polynomial upper bound $g(n)$. The algorithm with the self-tester has the following properties:

(i) if $\alpha_n^{-1} \leq g(n)$ (i.e., C2 holds), then the algorithm outputs a reliable numerical answer;

(ii) if $\alpha_n^{-1} > 4g(n)$ (i.e., C2 fails "badly"), then the algorithm outputs an error message;

[‡] Once again, we should point out that the analysis of theorem 5 refers to the idealized Markov chain in which all values β_i are exact. However, it is a simple matter to check that, assuming all stages are good, the effect on the mixing time of these small perturbations of the β_i is at most a constant factor.

(iii) if $g(n) < \alpha_n^{-1} \leq 4g(n)$ then the algorithm *either* outputs an error message *or* outputs a reliable numerical answer.

In every case the algorithm runs in polynomial time, and any numerical answer which is output is reliable with high probability.

Our method for computing the estimate $\widetilde{\alpha}_n$ again uses random sampling from the stationary distribution of the Markov chain M_{n-1}. The idea is to generate self-avoiding walks of lengths i and $n-i$ uniformly at random, and thus estimate the probability that such a pair can be glued together to form a walk of length n that is self-avoiding. The details are given in figure 2. Elementary statistics show that the sample size t required to ensure that the estimate $\widetilde{\alpha}_n$ is within the bounds specified above with high probability is a polynomial function of n, so the self-testing portion of the algorithm also runs in polynomial time.

Theorem 7 *The algorithm of section 2 with the self-tester incorporated runs in time polynomial in n, ϵ^{-1} and $\log \delta^{-1}$, and satisfies properties (i)-(iii) above.* □

It is worth noting that, while the primary motivation for the self-tester is to check the unverified conjecture C2, this idea could greatly increase the efficiency of the algorithm even if conjecture C2 were proven for some polynomial g. The self-tester is *computing* a close estimate for α_n, so that simulating the Markov chain for about $O(n^2\alpha_n^{-1})$ steps is sufficient to allow us to sample from close to the stationary distribution. This might be far more efficient than basing the number of simulation steps on some proven, but potentially loose, upper bound $g(n)$.

The idea of a tester has been used before, but in a much more restrictive sense. For example, Berretti and Sokal [2] propose testing possible "errors in scaling" due to the conjecture that $f(n) \approx An^{\gamma-1}$ by trying other specific polynomial forms for $f(n)$. This gives evidence that $f(n)$ might be of the correct form, but falls short of proving it probabilistically. In contrast, the tester we present is designed to verify exactly the conjecture we require, and therefore offers precisely quantified statistical evidence that our algorithm is operating as we expect.

4 Open questions

Our most obvious and compelling open problem is verifying conjecture C2. This would constitute a substantial breakthrough in the classical theory of self-avoiding walks. However, it is less well studied than conjecture C1, and its more elementary combinatorial nature should make this task more feasible. The results in this paper show that proving conjecture C2 for *any* polynomial g would yield the first provably polynomial time Monte Carlo approximation algorithms for self-avoiding walks.

Another direction is to find other natural problems that can be approached using the Monte Carlo techniques based on sub-Cayley trees described in this paper. For example, matchings (monomer-dimer coverings) in lattices can be uniformly generated using a Markov chain of this kind, and again the efficiency of the algorithm rests on a single combinatorial assumption. Unfortunately, however, unlike conjecture C2, in this case the analogous conjecture seems unlikely to be true. Nevertheless, perhaps it is possible to further adapt the algorithm so as to obtain a more reasonable conjecture.

Finally, we predict that there are other applications in which the type of self-testing described in this paper can be used to convert heuristics into robust algorithms. It would be interesting to explore the generality of this method for testing a conjecture in the region where it is sufficient to verify the correctness of an algorithm.

Acknowledgements

We thank Neal Madras and Alan Sokal for their valuable comments on an earlier version of this paper.

References

[1] Alm, S.E. Upper bounds for the connective constant of self-avoiding walks. *Combinatorics, Probability & Computing.* To appear.

[2] Berretti, A. and Sokal, A.D. New Monte Carlo method for the self-avoiding walk. *Journal of Statistical Physics* **40** (1985), pp. 483–531.

[3] Blum, M., Luby, M. and Rubinfeld, R. Self-testing/correcting with applications to numerical problems. *Proceedings of the 22nd ACM Symposium on Theory of Computing* (1990), pp. 73–83.

[4] Conway, A.R. and Guttmann, A.J. Lower bounds on the connective constant for square lattice self-avoiding walks. Preprint, 1992.

[5] Diaconis, P., and Stroock, D. Geometric bounds for eigenvalues of Markov chains. *Annals of Applied Probability* **1** (1991), pp. 36–61.

[6] Hammersley, J.M. The number of polygons on a lattice. *Proceedings of the Cambridge Philosophical Society* **57** (1961), pp. 516–523.

[7] Hammersley, J.M. On the rate of convergence to the connective constant of the hypercubical lattice. *Quarterly Journal of Mathematics, Oxford (2)* **12** (1961), pp. 250–256.

[8] Hammersley, J.M. and Morton, K.W. Poor man's Monte Carlo. *Journal of the Royal Statistical Society B* **16** (1954), pp. 23–38.

[9] Hammersley, J.M. and Welsh, D.J.A. Further results on the rate of convergence to the connective constant of the hypercubical lattice. *Quarterly Journal of Mathematics, Oxford (2)* **13** (1962), pp. 108–110.

[10] Hara, T., Slade, G. and Sokal, A.D. New lower bounds on the self-avoiding-walk connective constant. *Journal of Statistical Physics* **72** (1993), pp. 479-517.

[11] Jerrum, M. R. and Sinclair, A. J. Approximating the permanent. *SIAM Journal on Computing* **18** (1989), pp. 1149–1178.

[12] Jerrum, M.R., Valiant, L.G. and Vazirani, V.V. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science* **43** (1986), pp. 169–188.

[13] Karp, R., Luby, M. and Madras, N. Monte-Carlo approximation algorithms for enumeration problems. *Journal of Algorithms* **10** (1989), pp. 429–448.

[14] Lawler, G.F. *Intersections of Random Walks.* Birkhäuser, Boston, 1991.

[15] Lawler, G.F. and Sokal, A.D. Bounds on the L^2 spectrum for Markov chains and Markov processes: A generalization of Cheeger's inequality. *Transactions of the American Mathematical Society* **309** (1988), pp. 557–589.

[16] Madras, N. and Slade, G. *The Self-Avoiding Walk.* Birkhäuser, Boston, 1993.

[17] O'Brien, G.L. Monotonicity of the number of self-avoiding walks. *Journal of Statistical Physics* **59** (1990), pp. 969–979.

[18] Sinclair, A.J. *Algorithms for random generation and counting: a Markov chain approach.* Birkhäuser, Boston, 1992.

[19] Sinclair, A.J. Improved bounds for mixing rates of Markov chains and multicommodity flow. *Combinatorics, Probability & Computing* **1** (1992), pp. 351–370.

[20] Sinclair, A.J. and Jerrum, M.R. Approximate counting, uniform generation and rapidly mixing Markov chains. *Information and Computation* **82** (1989), pp. 93–133.

[21] Sokal, A.D. and Thomas, L.E. Exponential convergence to equilibrium for a class of random walk models. *Journal of Statistical Physics* **54** (1989), pp. 797–828.