Self-Testing Algorithms for Self-Avoiding Walks[†]

Dana Randall[‡] and Alistair Sinclair[§]

Abstract

We present a Markov chain 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 is shown rigorously to produce answers with specified accuracy and confidence. Only the efficiency of the algorithm relies on a widely believed conjecture, and a novel feature is that this conjecture can be *tested* as the algorithm proceeds. With this self-testing feature incorporated, the algorithm has polynomially bounded running time and is completely reliable, in the sense that it either outputs answers that are guaranteed to be within the specified accuracy and confidence bounds, or finds a counter-example to the conjecture.

1 Introduction

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 comprehensive survey, see the book of Madras and Slade [19]. (See also the book by Lawler [17] 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.

[†]A preliminary version of this paper appeared under the title "Testable Algorithms for Self-Avoiding Walks" in *Proceedings of the 5th SIAM/ACM Symposium on Discrete Algorithms*, 1994, pp. 593-602.

[‡]College of Computing and School of Mathematics, Georgia Institute of Technology, Atlanta GA 30332-0160. Email: randall@math.gatech.edu. Supported in part by NSF Career Grant No. CCR-970320.

[§]Computer Science Division, University of California, Berkeley CA 94720–1776. Email: sinclair@cs.berkeley.edu. Supported in part by NSF Grant CCR-9505448, by ICSI Berkeley, and by a UC Berkeley Faculty Research Grant.

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 $2 \le d \le 4$. 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 $2 \le d \le 4$.

One key fact that holds in all dimensions was discovered in 1954 by Hammersley and Morton [8]; they observed that $\lim_{n\to\infty} c_n^{1/n} = \mu$ exists, and that $\mu^n \le c_n = \mu^n f(n)$, 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., $\ell_{n+m} \le \ell_n + \ell_m$ 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

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

Here μ, A and γ are all dimension-dependent constants. The conjecture has in fact been proven for dimensions $d \ge 5$ by Hara and Slade [10, 11]. 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 for $d \le 4$, 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 [12] 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 many decimal places; for example, the most recent estimate for μ in two dimensions is $\mu = 2.63815852927 \pm 0.00000000001$ [13]. 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 [19] 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 (with no backtracking) 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 this edge is the last edge of the

walk, then it is removed; if the edge extends the walk so as to be self-avoiding, then it is added with probability β; 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 [26] prove 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 [18]). 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 the above 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 *a priori*, 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 [19] 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 Synopsis of 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 above. Our algorithm will have rigorously controlled statistical errors, and a running time that grows only slowly (i.e., as a low-degree polynomial) with the walk length and with the accuracy and confidence parameters. The following definitions make these notions precise; they are standard in the literature on efficient approximation algorithms for counting and uniform generation of combinatorial structures (see, e.g., [15, 16, 23]).

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 $\epsilon, \delta \in (0,1)$ (the *accuracy* and *confidence* parameters), 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 $\epsilon \in (0,1)$ (the *bias* parameter), outputs a self-avoiding walk of length n with

[†]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.

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 $\log \epsilon^{-1}$. □

Our algorithm will work with an increasing sequence $M_1, M_2, M_3, \ldots$ of Markov chains of the Berretti-Sokal type, where the state space of M_n consists of all self-avoiding walks of length at most n. We make three elementary but important innovations. First, we introduce a bootstrapping procedure whereby the number of steps required to simulate the nth Markov chain M_n is determined by an experiment performed using the previous Markov chain M_{n-1}. Second, we allow the parameter β in the Berretti-Sokal algorithm to vary at each level of the Markov chain (i.e., the transition probability β_n between walks of lengths $n-1$ and n now depends on n), and we *calculate* an appropriate value for β_n (which is first used in the chain M_n) from observations of the previous chain M_{n-1}. Thus we require no prior knowledge of β. These two innovations ensure the correctness of the algorithm without any assumptions; moreover, its running time will be polynomially bounded in n under a widely believed conjecture (C2 below) about self-avoiding walks. Our third innovation is a self-testing procedure which *guarantees* that the algorithm runs in polynomial time. The self-tester detects the validity of the conjecture in the region in which it is being assumed: as long as the test passes, the outputs are guaranteed to be reliable, while if the test fails we gain strong evidence that the widely-believed conjecture is incorrect. Either outcome is useful.

Ignoring the self-testing component for a moment, the behavior of our algorithm may be stated more precisely as follows. Fix a dimension d. 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, $\tilde{c}_n$ approximates c_n within ratio $(1+\epsilon)$ with probability at least $(1-\delta)$. In other words, the algorithm is a randomized approximation scheme as defined above. Moreover, once $\tilde{c}_n$ has been computed, we will have constructed a Markov chain M_n which can be used as an almost uniform generator for self-avoiding walks of length n.

The algorithm operates in stages, so that the estimate $\tilde{c}_n$ is computed using Markov chain M_{n-1}. The simulation length for the chain M_n (and hence the running time of the algorithm for walks of length n) depends polynomially on n, ϵ^{-1}, $\log \delta^{-1}$ and a natural quantity α_n associated with self-avoiding walks (see below for a definition). Actually the quantity α_n is not known analytically, but can be estimated using observations of the previous chain M_{n-1}. Thus the values of both α_n and β_n are bootstrapped in successive stages. A particularly attractive feature of our algorithm is that it is *interruptible*, in the sense that the values α_i, β_i determined during early stages of the algorithm can be reused at a future time without having to restart the program from scratch, should it be prematurely halted.

We now specify the quantity α_n, which plays a key role in our algorithm. For a fixed dimension d, define

$$\alpha_n = \min_{\substack{j,k \\ j+k \le n}} \frac{c_{j+k}}{c_j c_k}. \tag{1}$$

This quantity has the following natural interpretation. 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 independently and 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.

With this definition in place, we may now state the properties of the basic version of our algorithm.

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, so the linear dependence on α_n^{-1} in theorem 1 (see section 2.2) yields 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 a widely believed conjecture about self-avoiding walks, however, we may claim a much stronger bound on the running time. The conjecture in question is as follows: for a given dimension d, there exists a fixed polynomial g such that

$$c_j c_k \leq g(j+k)\, c_{j+k}, \quad \forall j,k. \tag{C2}$$

Thus in particular we have $\alpha_n^{-1} \leq g(n)$, i.e., if we choose random self-avoiding walks of lengths j and $k = n - j$ then the probability that their concatenation is self-avoiding is non-negligible (inverse polynomial in n). Conjecture C2 is no more restrictive than conjecture C1 of section 1.1, 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 $\frac{c_j c_k}{c_{j+k}} \sim A(\frac{jk}{j+k})^{\gamma-1} \leq A\left(\frac{j+k}{4}\right)^{\gamma-1}$. Thus conjecture C2 is also widely believed to hold, and is known to hold in dimensions $d \geq 5$ by the results of Hara and Slade mentioned earlier. Notice that conjecture C2 is in fact weaker than conjecture C1 since it makes no claims about the precise rate of growth of the function $\widetilde{f}(n)$. (As a simple example, suppose $\widetilde{f}(n) = n$ when n is odd and $\widetilde{f}(n) = n^2$ when n is even; then conjecture C2 would hold with $g(n) = \frac{n^2}{2}$ but conjecture C1 would fail.) 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 $\widetilde{f}$ from conjecture C1, with appropriate values for the constants μ, γ and A.

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. For dimensions $d \geq 5$, the same holds without any assumptions.* □

In the self-testing version of our algorithm, we incorporate a procedure that incrementally verifies conjecture C2 for successive values of n (for a specified polynomial g). Meanwhile, it assumes the correctness of the conjecture, but only for values of n *for which*

it has already been tested. This allows us to give an *a priori* polynomial bound on the running time of the algorithm so that *either* we will gather strong evidence (in the form of a counter-example) that the conjecture is false with the given polynomial g, *or* we will know that we can trust our simulations.

To make this more precise, fix a dimension d and a polynomial g, and suppose first that conjecture C2 holds for this g. Then, on inputs $\epsilon, \delta \in (0,1)$, the algorithm outputs a sequence of numbers $\widetilde{c}_1, \widetilde{c}_2, \widetilde{c}_3, \ldots$ and is a fully polynomial randomized approximation scheme, i.e., for each n, the time to output $\widetilde{c}_n$ is a polynomial function of n, ϵ^{-1} and $\log \delta^{-1}$ and, with probability at least $(1-\delta)$, $\widetilde{c}_n$ approximates c_n within ratio $(1+\epsilon)$. 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 has the above properties in the region previously explored (i.e., for $n < n_0$), but may be unreliable for larger values of n. However, in this case the algorithm will (with high probability) 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 properties of the self-tester are spelled out in more detail in theorem 7 of section 3.3. 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. This notion of *self-testing*, which either gives us confidence in our results or warns us that they may be erroneous, has been previously studied in the context of program checking (see, e.g., [3]).

The remainder of the paper is structured as follows. In section 2 we focus on the Markov chain M_n for a particular value of n, which lies at the heart of our algorithm. Our main task here will be to bound the rate of convergence of M_n to its stationary distribution. In section 3 we assemble the chains M_n for $n = 1, 2, 3, \ldots$ into our overall algorithm, whose running time depends on the quantity α_n; this will verify theorem 1 and corollary 2 above. Finally, in section 3.3 we show how to make the algorithm robust by adding a self-tester to verify conjecture C2, thus ensuring that the algorithm runs in time polynomial in n, ϵ^{-1} and $\log \delta^{-1}$ independently of any assumptions. We conclude by mentioning some open questions in section 4.

2 The Markov chain M_n

As indicated in section 1, 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 $\beta_1, \ldots, \beta_n \in (0,1)$, discussed below. In section 2.1 we define the chain M_n and deduce its basic properties, including its stationary distribution. We then go on to analyze its rate of convergence in section 2.2.

2.1 Definition and basic properties

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

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

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 $0, 1, \ldots, n$, with level i containing all walks of length i. 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 $\beta_i/2d$. In the case that $\beta_1 = \ldots = \beta_n$, this is a minor variant of the Markov chain used by Berretti and Sokal [2], but trunctated at level n.

For technical convenience, we in fact modify the Markov chain M_n by introducing a self-loop probability of $\frac{1}{2}$ at every state. I.e., at each step, M_n either (with probability $\frac{1}{2}$) makes a transition according to (2) above, or does nothing. Note that this modification merely injects a delay into the chain and does not affect its essential structure. We do this in order to make use of a convenient general result about mixing times (theorem 4 below). In practice, one would not need to introduce such a delay into the simulation.

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:

Proposition 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 (2). □

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 never greater than 1 was proven surprisingly recently by O'Brien [20].) 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 3 how to determine good approximations to the ideal values of β_i before they are needed. For the moment, we consider the behavior of the Markov chain assuming that each β_i is equal to c_{i-1}/c_i.

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

$$\pi_n(w) = \frac{1}{Z_n} \prod_{i=1}^{|w|} \frac{c_{i-1}}{c_i} = \frac{1}{Z_n c_{|w|}}. \tag{3}$$

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 1.2 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 equilibrium, 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 in 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 [14, 23]. 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

$$\rho(\Gamma) = \max_e \frac{1}{Q(e)} \sum_{\gamma_{xy} \ni e} \pi(x)\pi(y), \tag{4}$$

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* [25]. 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 [24] and is stated precisely in theorem 4 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

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

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 [24] *For an ergodic, reversible Markov chain with stationary distribution π and self-loop probabilities $P(y,y) \ge \frac{1}{2}$ for all states $y \in X$, we have*

$$\tau_x(\epsilon) \le \overline{\rho}\Big(\log \pi(x)^{-1} + \log \epsilon^{-1}\Big). \qquad \square$$

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 in (1). 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 (idealized) Markov chain M_n, starting at the empty walk $\mathbf{0}$, we have*

$$\tau_{\mathbf{0}}(\epsilon) \le K d n^2 \alpha_n^{-1}\Big(\log n + \log \epsilon^{-1}\Big)$$

for some constant K.

Proof. From (3) 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 $\rho(\Gamma) \le K' dn\alpha_n^{-1}$ 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 (4) becomes

$$\rho(\Gamma) = \max_e \; Q(e)^{-1}\pi_n(S)\pi_n(\overline{S}). \tag{5}$$

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. We have

$$Q(e) = \pi_n(w')P_n(w', w) = \frac{1}{4dZ_n c_{k+1}},$$

(where the extra factor of $\frac{1}{2}$ in $P_n(w', w)$ comes from the self-loops), and

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

where the first inequality follows from the sub-Cayley property of the tree. Putting these two calculations together, we see that $Q(e)^{-1}\pi_n(S)\pi_n(\overline{S}) \leq Q(e)^{-1}\pi_n(S) \leq 4dn\alpha_n^{-1}$. Since e was arbitrary, (5) 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 [26]. 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 [18], using the conductance (or Cheeger inequality). This latter proof is very similar in spirit to the one above; the main difference is that our proof replaces ρ^2 by $\overline{\rho}$ in the bound of theorem 4. □

3 The overall algorithm

In this section, we show how to assemble the sequence of Markov chains (M_n) 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 in section 1.3. 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 ingredients in the algorithm are two bootstrapping procedures for estimating the quantities α_n which appear in the mixing time and the parameters β_n governing the transition probabilities of the Markov chains. Recall that our analysis so far has assumed that we know α_n (the probability that two self-avoiding walks of total length at most n can

be glued together to form a new self-avoiding walk), and also 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 α_n and 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 α_n and β_n are required is in the Markov chain M_n. At the end of the section, we will show how to make the algorithm self-testing.

We stress that, throughout the paper, our goal is to sketch conceptually simple arguments that justify polynomial running times. We deliberately omit low-level details (in particular, constant factors) and make no attempt to optimize the time bounds. In any practical implementation of this algorithm, it would be necessary to refine the statistical procedures sketched here in order to obtain more practically useful bounds. We hope we have provided sufficient information for the interested reader to undertake this task. For an example of some tuning of this kind, see [21].

3.1 Bootstrapping α_n to determine the simulation time for M_n

In theorem 5 of section 2 we determined the mixing time of the nth Markov chain M_n as a function of n, ϵ, δ and the unknown quantity α_n. Thus we need to know at least a reasonable upper bound on α_n^{-1} in order to determine the number of simulation steps required for M_n. If we are prepared to accept conjecture 2, which asserts that $\alpha_n^{-1} \leq g(n)$ for some polynomial g, then we can simply substitute $g(n)$ for α_n^{-1} in the bound of theorem 5. However, we would like to have an algorithm which is independent of any conjectures. In this subsection we introduce a bootstrapping technique whereby we calculate an estimate $\tilde{\alpha}_n$ of α_n, such that $\alpha_n/4 \leq \tilde{\alpha}_n \leq \alpha_n$ with very high probability. Thus, from theorem 5, simulating M_n for $Kdn^2\tilde{\alpha}_n^{-1}(\log n + \log \epsilon^{-1})$ steps suffices to sample from close to its stationary distribution π_n, without appealing to any conjectures. Moreover, since $\tilde{\alpha}_n^{-1} \leq 4\alpha_n^{-1}$, the simulation time remains linear in α_n^{-1}.

Recall from (1) that $\alpha_n = \min_{j+k \leq n} \frac{c_{j+k}}{c_j c_k}$, and that for any fixed j and k this ratio is just the probability that two random self-avoiding walks of lengths j and k can be concatenated to form a new walk which is self-avoiding. Assuming inductively that we already know an upper bound on the mixing time of Markov chain M_{n-1}, we can use it to generate walks of each length $i \leq n-1$ (almost) uniformly. By sampling these smaller walks we can compute an estimate of α_n, thereby determining the number of steps for which we should simulate the next Markov chain M_n to guarantee that it is close to its stationary distribution.

This bootstrapping procedure, described in figure 1, works in detail as follows. Fix a dimension d. We assume first that we can generate walks of any given length $i < n$ (almost) uniformly at random, using Markov chain M_{n-1}, in time polynomial in n, α_{n-1} and $\log \epsilon^{-1}$ (where ϵ is the bias tolerated). This follows from the bound on the mixing time in theorem 5, and the fact that the stationary distribution π_{n-1} of M_{n-1} assigns equal weight to walks of each length $i < n$ and is uniform over lengths.[†] Thus to obtain t

[†]Actually the Markov chain we are simulating here is not precisely that analyzed in section 2, since the branching probabilities β_i will differ slightly from their ideal values. However, we know from proposition 3 that the stationary distribution is always uniform within each level of the tree, and it should also be clear that, assuming we control the errors in the β_i appropriately, the distribution over levels of the tree differs from the uniform distribution by at most a constant factor.

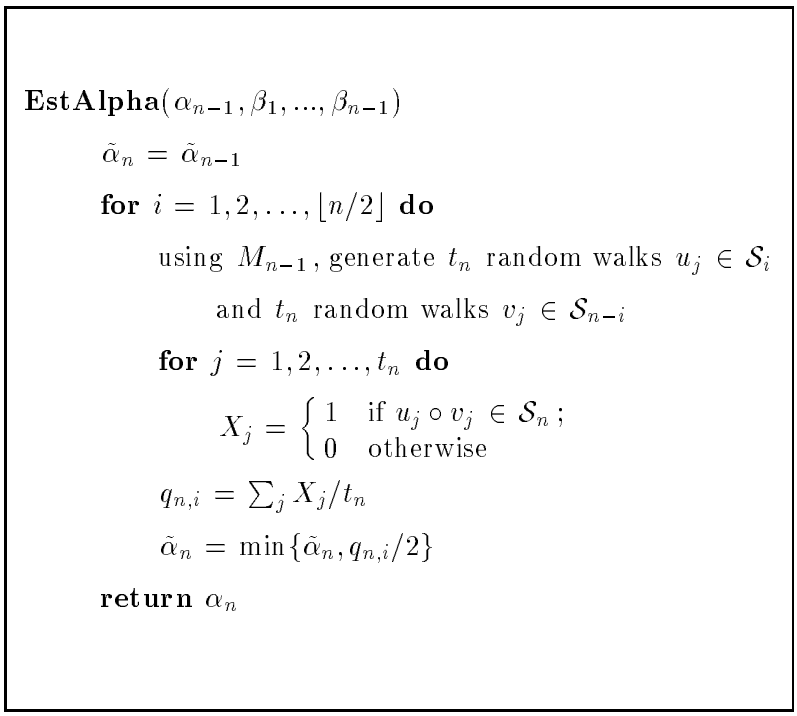

Figure 1: The subroutine for estimating α_n

independent random walks of length i, it is sufficient to generate $2nt$ independent samples from π_{n-1}; with high probability, at least t of these will have length i.[‡] For definiteness, we will assume that our algorithm aborts if at any point it fails to gather enough samples in such a procedure. The very small probability of this event can easily be absorbed into the confidence parameter δ.

We assume also that we have previously calcuated $\tilde{\alpha}_{n-1}$ such that $\alpha_{n-1}/4 \leq \tilde{\alpha}_{n-1} \leq \alpha_{n-1}$ (with high probability). The value $\tilde{\alpha}_{n-1}$ is an estimate of the probability that the concatenation of two self-avoiding walks of total length strictly smaller than n is self-avoiding, so we now need only estimate this probability for walks whose total length equals n. For each $0 < i < n$, we generate t_n independent pairs of walks of lengths i and $n-i$, and let $q_{n,i}$ be the proportion of these pairs whose concatenation is self-avoiding. Plainly $q_{n,i}$ is an (almost) unbiased estimator of the ratio $\frac{c_i c_{n-i}}{c_n}$. Now the 0/1 estimator theorem (see, e.g., [16]) tells us that $t_n = \mathrm{O}(\alpha_n^{-1} \log(n/\delta))$ samples suffice in order that $q_{n,i}$ is within a factor of 2 of this ratio with probability at least $1 - \delta/n^2$. Letting $\tilde{\alpha}_n = \min(\tilde{\alpha}_{n-1}, \min_i q_{n,i}/2)$, we get an estimate in the desired range $[\alpha_n/4, \alpha_n]$ with probability at least $1 - \delta$.[§]

The running time of EstAlpha is dominated by the time required to produce t_n samples

[‡]Notice that $2nt$ samples are sufficient to produce t samples of length i, even accounting for the fact that the distribution over levels of the tree is only approximately uniform.

[§]The loss of a factor $1/n^2$ in the confidence comes from the fact that $\tilde{\alpha}_n$ depends on $\Theta(n^2)$ independent random variables $q_{n,i}$. With more attention to detail, these errors could be controlled more efficiently.

of walks of each length $i < n$, using the Markov chain M_{n-1}. The total number of samples required is, with high probability, at most $O(t_n n^2)$ (actually $O(t_n n \log n)$), which from the above value of t_n is $O(n^2 \alpha_n^{-1}(\log n + \log \delta^{-1}))$. The time per sample is just (our estimate of) the mixing time of M_{n-1}, which we know inductively to be $O(n^2 \alpha_{n-1}^{-1}(\log n + \log \epsilon^{-1}))$ for any fixed dimension d, where ϵ is the bias from uniformity. Hence the overall running time of EstAlpha is $\tilde{O}(n^4 \alpha_n^{-2})$, where the $\tilde{O}$ notation suppresses both constant and logarithmic factors.

3.2 Estimating the branching probabilities β_n

Recall that the ideal value for the parameter β_n is the ratio c_{n-1}/c_n. Like α_n, this quantity may also be estimated by a bootstrapping procedure based on the Markov chain M_{n-1}. Note that in fact this ratio is precisely what one needs to compute c_n given c_{n-1}, so our estimate for β_n will immediately yield our estimate for c_n as well.

The overall algorithm is sketched in figure 2. The algorithm works in a sequence of stages, one for each successive value of n, corresponding to the iterations of the **for**-loop in figure 2. We call stage n *good* if it computes a value β_n that approximates the value c_{n-1}/c_n within ratio $(1 + \epsilon/4n^2)$, where ϵ is the accuracy input.

CountSAWs

$\beta_1 = 1/2d$; $\tilde{c}_1 = 2d$; $\alpha_1 = 1$

for $n = 2, 3, 4, \ldots$ **do**

using M_{n-1}, generate T_n random walks $w_j \in \mathcal{S}_{n-1}$

let $E = \sum_j |\{w \in \mathcal{S}_n : w_j \prec_1 w\}|$

set $\beta_n = T_n/E$

output $\tilde{c}_n = \tilde{c}_{n-1}/\beta_n$

let α_n=**EstAlpha**$(\alpha_{n-1}, \beta_1, ..., \beta_{n-1})$

Figure 2: The overall algorithm

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 any 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 $c_n \geq c_{n-1}$). The sample size is controlled by the parameter T_n. A simple generalization of the 0/1-estimator theorem (see

[16]) 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. Specifically, we get:

Lemma 6 *In the algorithm of figure 2, 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 our algorithm, run to stage n, satisfies the requirements of a randomized approximation scheme for c_n (see Definition (i)), which was one of our principal goals. Moreover, by the end of stage n we have computed values β_i for $1 \leq i \leq n$; thus we have constructed a Markov chain M_n which we can simulate to produce an almost uniform generator for self-avoiding walks of any length up to n (see Definition (ii)). This was our second principal goal.

Ignoring EstAlpha for a moment, the running time of stage n of the algorithm is dominated by the time required to produce T_n (almost) uniform random self-avoiding walks of length $n-1$ using Markov chain M_{n-1}. As in the analysis of EstAlpha in the previous subsection, we can bound this by $O(T_n n^2 \alpha_{n-1}^{-2}(\log n + \log \epsilon^{-1}))$ for any fixed dimension d.¶ Plugging in the bound on T_n from lemma 6, we deduce that the total running time of stage n is polynomial in n, α_n^{-1}, ϵ^{-1} and $\log \delta^{-1}$. This is true also for the call to EstAlpha, as we saw in the previous subsection. Thus the approximation scheme for c_n has the properties claimed in theorem 1. By the same reasoning, simulating the Markov chain M_n for $O(n^2 \tilde{\alpha}_n^{-1}(\log n + \log \epsilon^{-1}))$ steps can be used as the basis for an almost uniform generator of walks of length n with the properties claimed in theorem 1.

This algorithm is particularly well suited to applications where many samples of self-avoiding walks of a given length are required. In order to produce just one sample, a fair amount of work is required in order to determine estimates for the β_i and α_i. However, once this work is done, the Markov chain M_n can be used to generate as many samples as desired much more efficiently. It is worth noting that in this sense the algorithm is interruptible; as long as the estimates for each β_i and α_i are stored (to specified accuracy and confidence), this initial work does not have to be repeated.

3.3 Making the algorithm self-testing

In the previous subsection we presented an algorithm whose correctness is independent of any conjectures; however, to get an *a priori* bound on the running time of the algorithm we must appeal to conjecture C2. In this section we show how to place the algorithm on a firmer theoretical footing by algorithmically testing conjecture C2, thus guaranteeing that

¶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.

the algorithm runs in polynomial time. 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 conjecture C2 states that, for all j and k,

$$c_j \; c_k \leq g(j+k) \; c_{j+k}, \tag{6}$$

where g is some fixed polynomial. The point of this conjecture is that it immediately implies $\alpha_n^{-1} \leq g(n)$ for all n, which in turn gives us a polynomial upper bound on the mixing time of the Markov chain M_n.

Clearly, to argue about the chain M_n it is sufficient to establish condition (6) only for $j + k \leq n$. Thus, provided we can test the condition in this range using only information from the previous stage $n-1$, we can be sure that stage n is reliable also. As it turns out, our procedure for calculating $\tilde{\alpha}_n$ (figure 1) at the end of stage $n-1$ makes testing the conjecture in the above range almost immediate. Recall that in section 3.1 we derived statistical guarantees stating that $\alpha_n/4 \leq \tilde{\alpha}_n \leq \alpha_n$ with very high probability. Therefore, if $\alpha_n^{-1} \leq g(n)$, then our estimate must satisfy $\tilde{\alpha}_n^{-1} \leq 4g(n)$ with high probability. Conversely, if our estimate satisfies this condition then we can assert with high confidence that $\alpha_n^{-1} \leq 4g(n)$. Thus our testing procedure, spelled out in figure 3.3, simply amounts to comparing $\tilde{\alpha}_n^{-1}$ with $4g(n)$. In practice we could use for $g(n)$ either the polynomial implied by conjecture C1 (which is essentially just $\widetilde{f}$, with a slightly different constant A) or, to give ourselves a bit more room, a slightly larger polynomial. Note that our choice of g will affect the running time, since we will be substituting $g(n)$ in place of α_n^{-1} in our previous analysis.

SelfTest

let α_n = **EstAlpha**$(\alpha_{n-1}, \beta_1, ..., \beta_{n-1})$

if $\tilde{\alpha}_n^{-1} > 4g(n)$

then output "Warning: conjecture fails"

else continue

Figure 3: The self-tester

Summing up the above discussion, we have:

Theorem 7 *The algorithm of figure 2 with the self-tester incorporated runs in time polynomial in n, ϵ^{-1} and $\log \delta^{-1}$. Furthermore, assuming that no warning has been issued at any stage $i < n$, it satisfies the following properties:*

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

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

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

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

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 for dimensions $d \leq 4$. 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 (even, say, for $g(n) = An^{100}!$) 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, and an anonymous referee for several useful suggestions.

References

[1] Alm, S.E. Upper bounds for the connective constant of self-avoiding walks. *Combinatorics, Probability & Computing* **2** (1993), pp. 115–136.

[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 bound on the connective constant for square lattice self-avoiding walks. *Journal of Physics A* **26** (1993), pp. 3719–3724.

[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 latti ce. *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. Self-avoiding walk in five or more dimensions. I. The critical behaviour. *Communications in Mathematical Physics* **147** (1992), pp. 101–136.

[11] Hara, T., Slade, G. The lace expansion for self-avoiding walk in five or more dimensions. *Reviews in Mathematical Physics* **4** (1992), pp. 235–327.

[12] 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.

[13] Jensen, I. and Guttmann, A. J. Self-avoiding polygons on the square lattice. *Journal of Physics A* **32** (1999), pp. 4867–4876.

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

[15] 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.

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

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

[18] 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.

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

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

[21] Randall, D. *Counting in lattices: combinatorial problems from statistical mechanics.* PhD Thesis, UC Berkeley, Fall 1995. Also available as Technical Report TR-94-055, International Computer Science Institute, Berkeley.

[22] Randall, D. and Sinclair, A.J. Testable algorithms for self-avoiding walks. *Proceedings of the 5th ACM/SIAM Symposium on Discrete Algorithms* (1994), pp. 593-602.

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

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

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

[26] 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.