Chapter 9

Analysis of Monte Carlo methods

9.1 Fundamentals and basic examples

Monte Carlo methods are useful for getting statistical estimates on the values of the connective constant, critical exponents, and other quantities related to self-avoiding walks. Essentially, a Monte Carlo simulation is a computer experiment which observes random versions of a particular system. After we obtain enough data, we can use statistical techniques to get estimates and confidence intervals for the desired quantities.

For definiteness, consider the exponent ν [defined in (1.1.5)], which measures the length scale of self-avoiding walks. There are several unresolved questions about ν, such as: Are the conjectured values (1.1.12) and (1.1.14) correct in 2, 3, and 4 dimensions? In particular, is the Flory exponent 3/5 too large in three dimensions? Do the hyperscaling relations (1.4.14) and (1.4.24) hold? In two dimensions, does the average area enclosed by an N-step self-avoiding polygon scale like $N^{2\nu}$? Good numerical estimates can give evidence in support of (or against) these and other conjectures. As we saw in Section 2.3, such evidence can also be relevant for analogous conjectures in other models; for example, if hyperscaling fails for self-avoiding walks in three dimensions, then it is likely to fail for other N-vector models as well.

To get a taste of some of the numerical values that various researchers have obtained, let us focus on the value of ν in three dimensions. An early study by Rosenbluth and Rosenbluth (1955) used *biased sampling* (see Section 9.3.1) to generate walks of up to 64 steps, obtaining an esti-

N. Madras and G. Slade, *The Self-Avoiding Walk*,
Modern Birkhäuser Classics, DOI 10.1007/978-1-4614-6025-1_9,

mate of 0.61 for ν. Stellman and Gans (1972) generated walks of up to 298 steps using a continuum version of the *pivot algorithm* (see Section 9.4.3) to obtain an estimate of 0.610 ± 0.008 for ν (this and the following are *95% confidence intervals* for ν; see Section 9.2.1). Grishman (1973) generated walks of length 500 using a combination of the *dimerization* and *enrichment* algorithms (see Sections 9.3.2 and 9.3.3), producing an estimate of 0.602 ± 0.009. However, these early results, which used relatively short walks, are biased by significant systematic errors due to unincluded correction-to-scaling terms (see Section 9.2.1). Rapaport (1985) generated walks of length up to 2400 using a combination of dimerization and enrichment, and estimated 0.592 ± 0.004. Madras and Sokal (1988) used the pivot algorithm to generate walks of up to 3000 steps, and obtained 0.592 ± 0.003. A very recent study (Li and Sokal, private communication), which uses the pivot algorithm to generate walks of up to 80,000 steps, indicates that the true value of ν is even lower: the preliminary estimate is 0.5883 ± 0.0013, which is in remarkable agreement with the field theoretic renormalization group prediction of 0.5880 ± 0.0015 obtained by Le Guillou and Zinn-Justin (1989). This brief history illustrates that correction-to-scaling terms are a serious danger, and that exponent estimates based on short walks must be interpreted with caution.

There are good reasons why Monte Carlo is "easier" for self-avoiding walks than for spin systems. First, there is only one limit to worry about, namely the length of the walk going to infinity. In a spin system, one has to take a limit going to a critical temperature as well as a thermodynamic limit of a finite lattice increasing to $\mathbf{Z}^d$. The latter is absent for self-avoiding walks, which can be simulated without any errors arising from the finite volume of the lattice. Secondly, spin system simulations typically exhibit "critical slowing-down": as the correlation length ξ diverges, you must look at finite lattices of at least ξ^d sites to learn anything, and you must look at each site before you get a new data point. This is not an inherent restriction for self-avoiding walks, since you only have to look at sites occupied by the walk. This suggests the possibility of more efficient algorithms in which critical slowing-down is much less severe.

Another frequently used numerical method is exact enumeration and extrapolation. This approach computes exact values of certain quantities for small values of N and then tries to infer an asymptotic behaviour from these numbers. We will not discuss this method in this book; the interested reader is referred to Guttmann (1989a).

To conduct a Monte Carlo experiment for the estimation of ν, one can for example proceed as follows.

(a) Select several values of N, say $N_1, \ldots, N_m$.

(b) For each N_i, generate many N_i-step self-avoiding walks at random. Use these to get an estimate $\hat{Y}_i$ of $\langle |\omega(N_i)|^2 \rangle$, along with an estimate of the uncertainty in $\hat{Y}_i$.

(c) Fit a curve of the form $Y = AN^{2B}$ through the points $(N_i, \hat{Y}_i)$. The "best" value of B will be the estimate of ν.

Of course, each step raises many questions about how to proceed. In (a), how many and which values of N should be chosen? In (b), how many is "many"? What is the most efficient way to generate walks at random? How can the uncertainty best be estimated, and how does this uncertainty vary with N? In (c), how do we use the estimated uncertainties to fit data to a curve that is only believed to be *asymptotically* correct? These are the kinds of question that will be addressed in this chapter. We shall concentrate, however, on what one can say *rigorously* about the properties of these methods. The reader who wishes to pursue other aspects of Monte Carlo in more depth should consult the references listed in the Notes at the end of this chapter.

The remainder of this section will discuss some basic examples of Monte Carlo methods for generating self-avoiding walks, and will use them to illustrate various themes that appear throughout the chapter. Section 9.2 focuses on some statistical aspects—both practical and theoretical—of Monte Carlo methods. Sections 9.3 through 9.6 will treat various methods in detail. The longer proofs and calculations are deferred to Section 9.7.

We use our usual notation that $\mathcal{S}_N$ is the set of all N-step self-avoiding walks that begin at the origin. We shall restrict our attention to walks that begin at the origin, unless explicitly stated otherwise.

We begin with the basic question: How can we choose an N-step self-avoiding walk at random? (In this context, "at random" means that all walks in $\mathcal{S}_N$ are equally likely. For now, N is a given integer.) One simple method is the following:

Elementary Simple Sampling (ESS). This algorithm generates ordinary simple random walks until it obtains an N-step walk that is self-avoiding.

1. Let $w(0)$ be the origin and set $i = 0$.
2. Increase i by one. Choose one of the $2d$ neighbours of $w(i-1)$ at random, and let $w(i)$ be that point.
3. If $w(i) = w(j)$ for some $j = 0, 1, \ldots, i-1$, then go back to Step 1. Otherwise, go to Step 2 if $i < N$, and stop if $i = N$.

When this algorithm terminates, the walk $W \equiv (w(0), \ldots, w(N))$ is self-avoiding. Moreover, we claim that for any $\omega \in \mathcal{S}_N$ we have $\Pr\{W = \omega\} =$

$1/c_N$. To see this, let $\mathcal{S}_N^\circ$ be the set of all N-step (ordinary) simple walks. If we keep choosing members of $\mathcal{S}_N^\circ$ uniformly at random until one of them is in $\mathcal{S}_N$, then the final result is evidently uniformly distributed on $\mathcal{S}_N$. But this is essentially what the above algorithm does; Step 3 is just a short-cut to avoid generating the last $N - i$ steps of a walk that we already know intersects itself by the i-th step. Thus the ESS algorithm indeed generates a self-avoiding walk at random. However, it can be very slow when N is even moderately large: the probability that an N-step simple random walk is self-avoiding is $c_N/(2d)^N$, so the expected number of attempts (i.e. returns to Step 1) is $(2d)^N/c_N$. Therefore, using the notation T_X to represent the expected amount of computer time required for algorithm X to generate a single N-step self-avoiding walk, we have

$$T_{ESS} = \left(\frac{2d}{\mu}\right)^{N+o(N)} \tag{9.1.1}$$

We can improve on the efficiency of ESS by only generating simple random walks with no immediate reversals, as follows:

Non-Reversed Simple Sampling (NRSS). This algorithm generates simple random walks with no immediate reversals until it obtains an N-step walk that is self-avoiding.

1. Let $w(0)$ be the origin. Choose one of the $2d$ neighbours of the origin at random, and let $w(1)$ be that point. Set $i = 1$.
2. Increase i by one. Of the $2d - 1$ neighbours of $w(i-1)$ that are different from $w(i-2)$, choose one at random, and let $w(i)$ be that point.
3. If $w(i) = w(j)$ for some $j = 0, 1, \ldots, i-1$, then go back to Step 1. Otherwise, go to Step 2 if $i < N$, and stop if $i = N$.

Arguing as for the ESS algorithm, we see that the NRSS algorithm generates a self-avoiding walk uniformly from $\mathcal{S}_N$, and it takes an average of $2d(2d-1)^{N-1}/c_N$ attempts to do so. Therefore

$$T_{NRSS} = \left(\frac{2d-1}{\mu}\right)^{N+o(N)}, \tag{9.1.2}$$

which is better than (9.1.1), but still not very good.

Before continuing, we should mention the following "obvious" algorithm, which (perhaps surprisingly at first sight) *does not work*:

Myopic Self-Avoiding Walk (MSAW). Execute a random walk, at each step choosing only from those sites that have not yet been visited.

1. Let $w(0)$ be the origin, and set $i = 0$.
2. Increase i by one. Of the neighbours of $w(i-1)$ that are not in the set $\{w(0), \ldots, w(i-2)\}$, choose one at random, and let $w(i)$ be that point. (If all of the neighbours of $w(i-1)$ are in this set, then the walk is trapped, so return to Step 1.)
3. Repeat Step 2 if $i < N$, and stop if $i = N$.

This algorithm produces a walk in $\mathcal{S}_N$, but with the *wrong distribution*. To see where the problem is, consider four-step walks on $\mathbf{Z}^2$: the probability of obtaining the walk NEEE on a given attempt is $\frac{1}{4} \times \frac{1}{3} \times \frac{1}{3} \times \frac{1}{3}$, but the probability of obtaining the walk NESE is $\frac{1}{4} \times \frac{1}{3} \times \frac{1}{3} \times \frac{1}{2}$. Thus, the probabilities are not uniform on $\mathcal{S}_N$. In fact, the probabilities become very far from uniform for large N. The algorithm MSAW actually defines a different model, which is essentially the same as the "true self-avoiding walk" of Section 10.4.

Other algorithms for generating independent self-avoiding walks are described in Section 9.3. To varying degrees, they all suffer from the problem that they are inefficient for large walks. In fact we have the following

> **Open Problem:** Is there an algorithm A which generates a single N-step self-avoiding walk, with distribution that is exactly uniform on $\mathcal{S}_N$, such that the average time T_A is bounded by a polynomial in N?

Actually, the problem is only open in low dimensions: for $d \geq 5$, the average time of the dimerization algorithm of Section 9.3.2 is known to be bounded by a polynomial. Dimerization is also the most efficient known algorithm for generating a single walk exactly uniformly in any dimension, with an expected running time of $N^{O(\log N)}$ (if the usual scaling assumptions are true; see Section 9.3.2). However, there do exist more efficient algorithms that generate self-avoiding walks with a distribution that is arbitrarily close to uniform. These algorithms do not attempt to generate a sequence of independent self-avoiding walks, but rather they use a *Markov chain* to generate a sequence of self-avoiding walks that is not independent. Such methods are known as *dynamic*[1], as opposed to *static*, Monte Carlo methods. Roughly speaking, dynamic methods generate new walks by modifying (or "updating") walks that have been previously generated, while static methods build up walks from scratch. Static methods yield independent walks (or independent groups of walks), while dynamic methods yield correlated sequences of walks.

[1] This usage is distinct from the term "polymer dynamics", which refers to the (real) motion of polymers.

The basic idea of the dynamic approach is the following. Suppose that π is a probability distribution on some set S (i.e., for each $i \in S$, $\pi(i)$ is the probability of i, and $\sum_{i \in S} \pi(i) = 1$), and that we wish to generate a random object with the distribution π. If we can find a Markov chain with state space S whose unique equilibrium distribution is π, then the fundamental theory of Markov chains tells us that running this chain for a long time will produce observations whose distribution approaches π. In our case, we may take $S = \mathcal{S}_N$ and $\pi(\omega) = 1/c_N$ for every self-avoiding walk ω in $\mathcal{S}_N$. We begin with a walk $\omega^{[0]}$ in $\mathcal{S}_N$ and apply some (randomized) procedure that changes $\omega^{[0]}$ to get another self-avoiding walk $\omega^{[1]}$; then we apply the same procedure to $\omega^{[1]}$ to get another walk $\omega^{[2]}$, and so on. In this way we generate a sequence of walks $\{\omega^{[n]} : n \geq 0\}$ such that (for sufficiently large n) the distribution of $\omega^{[n]}$ is arbitrarily close to π. This sequence of walks will be correlated, of course, but one hopes that the relevant correlations will decay quickly.

To make the preceding discussion more precise, we make the following definitions, which are fairly standard in probability textbooks:

Definition 9.1.1 *Let $\{X^{[t]} : t = 0, 1, \ldots\}$ be a Markov chain on a finite or countably infinite state space S. Let*

$$P(i,j) = \Pr\{X^{[t+1]} = j | X^{[t]} = i\} \qquad (t \geq 0, i, j \in S)$$

be the one-step transition probabilities of the chain, and for every nonnegative integer n let

$$P^n(i,j) = \Pr\{X^{[t+n]} = j | X^{[t]} = i\} \qquad (t \geq 0, i, j \in S)$$

be the n-step transition probabilities. (We only consider chains that are time-homogeneous, i.e. whose transition probabilities are independent of t.) The chain is said to be irreducible *if for every i and j in S there exists an $n > 0$ such that $P^n(i,j) > 0$ (i.e. every state can be reached from every state). An irreducible chain is said to have period p if p is the greatest common divisor of $\{n : P^n(i,i) > 0\}$ for every state i (or equivalently for at least one i). A chain which has period 1 is said to be* aperiodic.

We remark that P^n is simply the n-th matrix power of P. Notice that if an irreducible chain has $P(i,i) > 0$ for some i, then it is aperiodic.

The standard theory of Markov chains (see references in Notes) tells us the following about the long-term behaviour of an aperiodic irreducible chain $X^{[t]}$. First, the limit

$$\lim_{n \to \infty} P^n(i,j)$$

exists for every i and j in S, and this limit is independent of i; call it $\pi(j)$. Next, if S is finite, then

$$\sum_{j \in S} \pi(j) = 1 \tag{9.1.3}$$

and

$$\sum_{i \in S} \pi(i)P(i,j) = \pi(j) \quad \text{for every } j \text{ in } S; \tag{9.1.4}$$

and moreover π is the *only* nonnegative solution of (9.1.3) and (9.1.4). Finally, if S is countably infinite, then there are two possibilities: either $\pi(j) = 0$ for every j, in which case (9.1.3) and (9.1.4) have no nonnegative solution; or else π is the unique solution of (9.1.3) and (9.1.4).

An important special case is the following: we say that a chain is *reversible* with respect to π if

$$\pi(i)P(i,j) = \pi(j)P(j,i) \quad \text{for every } i \text{ and } j \text{ in } S. \tag{9.1.5}$$

(In alternate terminology, (9.1.5) is called the *detailed balance condition.*) Note that if π is the uniform distribution, then reversibility is equivalent to symmetry of P. If a chain is reversible with respect to π, then (9.1.4) holds (to see this, sum (9.1.5) over i). In practice, almost all dynamic Monte Carlo procedures use reversible chains (or are a combination of several reversible chains, as in Section 9.5.2).

If (9.1.3) and (9.1.4) hold for an irreducible chain and some π, then the chain is said to be *positive recurrent*, and π is called its *equilibrium*, or *stationary, distribution.* In general, $\pi(j)$ is the fraction of time that the chain spends in state j, in the long run (irrespective of the initial state). Thus, if our chain $X^{[t]}$ is positive recurrent, and if we observe it for a sufficiently long time, then the data should be pretty representative of the distribution π. For example, this tells us that if the state space is $\mathcal{S}_N$ and we observe end-to-end distance of the walks $X^{[t]}$ for a sufficiently long time, then we will obtain a good estimate of the mean square displacement $\langle |\omega(N)|^2 \rangle$ computed according to π. This is essentially a consequence of the ergodic theorem, which tells us that for a real-valued function f on the state space of a positive recurrent chain,

$$\lim_{m \to \infty} \frac{1}{m} \sum_{t=1}^{m} f(X^{[t]}) = \sum_{i \in S} f(i)\pi(i)$$

with probability one (assuming that the right hand side, which is just the expectation of $f(\cdot)$ with respect to π, is absolutely convergent).

Let us now look at a particular example: an algorithm due to Verdier and Stockmayer (1962) which turns one self-avoiding walk into another by

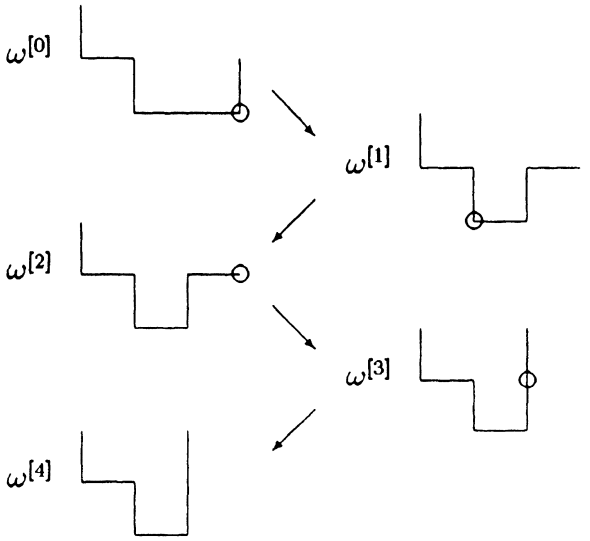

Figure 9.1: An example of the Verdier-Stockmayer algorithm in action. The circled site of $\omega^{[t]}$ corresponds to the randomly chosen I of Step 2. Observe that $\omega^{[2]} = \omega^{[1]}$ because the $\tilde{\omega}$ resulting from $\omega^{[1]}$ is not self-avoiding. Also observe that the $\tilde{\omega}$ resulting from $\omega^{[3]}$ in fact equals $\omega^{[3]}$.

moving one or two bonds of the walk. Briefly, it picks a site at random and tries to "flip" the two incident bonds if they form a right angle (or tries to wiggle the end bond if the chosen site is an endpoint of the walk). Here is a precise statement of the algorithm; a verbal description follows, and Figure 9.1 gives an illustration.

Verdier-Stockmayer (V-S) Algorithm. This algorithm generates a Markov chain $\{\omega^{[t]} : t = 0, 1, \ldots\}$ on the state space $\mathcal{S}_N$ which is reversible with respect to the uniform distribution on $\mathcal{S}_N$.

1. Let $\omega^{[0]}$ be any self-avoiding walk in $\mathcal{S}_N$. Set $t = 0$.
2. Choose an integer I uniformly at random from $\{0, 1, \ldots, N\}$.
3. Define a new walk $\tilde{\omega} = (\tilde{\omega}(0), \ldots, \tilde{\omega}(N))$, which is not necessarily self-avoiding, as follows. First set $\tilde{\omega}(l) = \omega^{[t]}(l)$ for all $l \neq I$. Then:

(a) if $0 < I < N$, then set $\tilde{\omega}(I) = \omega^{[t]}(I-1) +$

$(\omega^{[t]}(I+1) - \omega^{[t]}(I))$;

(b) if $I = N$, then set $\tilde{\omega}(N)$ equal to any neighbour of $\omega^{[t]}(N-1)$ except for $\omega^{[t]}(N-2)$ and $\omega^{[t]}(N)$, chosen at random;

(c) if $I = 0$, then set $\tilde{\omega}(0)$ equal to any neighbour of $\omega^{[t]}(1)$ except for $\omega^{[t]}(0)$ and $\omega^{[t]}(2)$, chosen at random. Then translate $\tilde{\omega}$ so that it begins at the origin.

4. If $\tilde{\omega}$ is self-avoiding, then set $\omega^{[t+1]} = \tilde{\omega}$; otherwise, set $\omega^{[t+1]} = \omega^{[t]}$.
5. Increase t by one and go to Step 2.

To visualize this algorithm, think of the N bonds of a walk ω as a sequence of N unit vectors $\Delta\omega(i) \equiv \omega(i) - \omega(i-1)$ $(i = 1, \ldots, N)$. Step 2 chooses a site $\omega(I)$ at random. Then Step 3 either interchanges the I-th bond with the $(I+1)$-th bond (if $0 < I < N$) or else randomly changes the first or last bond (if I is 0 or N). [Observe that in Step 3(a) we obtain $\Delta\tilde{\omega}(I) = \Delta\omega(I+1)$ and $\Delta\tilde{\omega}(I+1) = \Delta\omega(I)$.] Step 4 rejects the proposed walk ω if it is not self-avoiding.

To show that a certain probability distribution π is the equilibrium distribution of a Markov chain, we check both reversibility and irreducibility. First we shall show that the V-S algorithm is reversible (with respect to the uniform measure on $\mathcal{S}_N$). To do this, it suffices to check that P is symmetric, i.e.

$$P(\omega, \omega') = P(\omega', \omega) \qquad \text{whenever } \omega \neq \omega'. \tag{9.1.6}$$

So suppose that ω and ω' are distinct walks in $\mathcal{S}_N$. If $P(\omega, \omega') = 0$ and $P(\omega', \omega) = 0$, then (9.1.6) holds, so assume without loss of generality that $P(\omega, \omega') > 0$. That is, if we start with ω, then there is a choice of I such that the walk $\tilde{\omega}$ obtained in Step 3 equals ω'. In this case, ω and ω' differ by either one or two bonds, and so there is a *unique* choice of I that transforms ω into ω'; denote this unique number by $i[\omega, \omega']$. Thus, since $\Pr\{I = i[\omega, \omega']\} = 1/(N+1)$, we have

$$P(\omega, \omega') = \begin{cases} \frac{1}{N+1} & \text{if } 0 < i[\omega, \omega'] < N \\ \frac{1}{(N+1)(2d-2)} & \text{if } i[\omega, \omega'] \text{ is 0 or } N \end{cases}$$

(the second line follows since there are $2d-2$ ways to choose the new first or last bond). Now, if ω can be transformed into ω', then ω' can be transformed into ω; in particular, we have $i[\omega', \omega] = i[\omega, \omega']$. Thus $P(\omega', \omega)$ is given by the right hand side of the above equation, and so (9.1.6) holds.

There is a subtlety in the algorithm that makes reversibility so easy to prove. Consider a variation of the V-S algorithm in which we wait until a successful move occurs before recording the next observation: specifically, suppose that Steps 4 and 5 are replaced by

> 4′. If $\tilde{\omega}$ is not self-avoiding, then go to Step 2. If $\tilde{\omega}$ is self-avoiding, then set $\omega^{[t+1]} = \tilde{\omega}$, increase t by one, and go to Step 2.

Now there is no guarantee that (9.1.6) holds; the proof fails because in the new chain the one-step transition probability from ω to ω' is $P(\omega,\omega')/(1 - P(\omega,\omega))$ (here P refers to the probabilities in the original chain; observe that $1 - P(\omega,\omega)$ is the probability that a single attempt turns ω into something different). So we cannot have symmetry in the new chain unless $P(\omega,\omega)$ is the same for every ω in the original chain. This does not happen for the V-S algorithm, nor for any other interesting algorithm that we know of. Thus we see that in order to guarantee that we get the correct equilibrium distribution, it is vital to record the current walk *after every attempt*, whether the attempt results in a walk that is self-avoiding (a "success") or not (a "rejection").

We have seen that the original V-S algorithm is reversible, but unfortunately it is *not* irreducible. For example, there exist self-avoiding walks which are "frozen", i.e. they can never be changed by the V-S algorithm (see Figure 9.2). But the irreducibility difficulties are worse than just

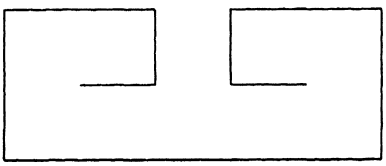

Figure 9.2: A 17-step self-avoiding walk in $\mathbf{Z}^2$ which is "frozen" with respect to the Verdier-Stockmayer algorithm.

having a few frozen walks. An *ergodicity class* of a Markov chain is defined to be a maximal subset A of the state space such that for every i and j in A, there exists a $n > 0$ such that $P^n(i,j) > 0$. Thus $\mathcal{S}_N$ is partitioned into many ergodicity classes, some of which contain a single walk. If $\omega^{[0]}$ is in a given ergodicity class, then we can view the V-S algorithm as producing a Markov chain whose equilibrium distribution is uniform *on that ergodicity class*, not on all of $\mathcal{S}_N$. As we shall see, this is a serious concern in principle,

because the largest of the ergodicity classes is an exponentially small part of $\mathcal{S}_N$ as $N \to \infty$ (Theorem 9.4.2).

We conclude this introductory section with some remarks on the following problem, which is relevant to any computer program that works with self-avoiding walks: how fast can we check that a given walk is self-avoiding? To be precise, suppose that you are given a finite sequence of lattice sites $\omega(0), \omega(1), \ldots, \omega(N)$ such that $|\omega(i) - \omega(i-1)| = 1$ for every $i = 1, \ldots, N$. What is the most efficient way to check whether these $N+1$ sites are all distinct?

The most obvious algorithm is to look at every pair i and j such that $0 \le i < j \le N$ and check whether $\omega(i)$ equals $\omega(j)$. There are $N(N+1)/2$ such pairs, so the running time of this algorithm is $O(N^2)$. A different algorithm achieves a running time of $O(N)$ by using a "bit map". The idea behind this method is to simply draw a picture of the walk. For example, suppose we are working with N-step walks starting at the origin in $\mathbf{Z}^2$. The simplest bit map is a $(2N+1) \times (2N+1)$ array, indexed by (i,j), $-N \le i,j \le +N$, with all entries initially 0. Then every site of $\mathbf{Z}^2$ that can be reached by an N-step walk corresponds to an entry. For each $i = 0, 1, \ldots, N$ in turn, check the entry corresponding to the site $\omega(i)$: if the entry is 0 then change it to 1, but if the entry is already 1 then the walk is not self-avoiding. Afterward, go through the list of sites again to reset the entries to 0. The running time of this algorithm is clearly $O(N)$.

The disadvantage of a bit map is that it requires a lot of space: in $\mathbf{Z}^d$, it requires $O(N^d)$ words of computer memory. An alternative approach uses a data structure known as a "hash table". A set of N sites can be stored in a hash table of size $O(N)$ in such a way that we can check whether a given site is in the set in *average* time $O(1)$ — i.e. independent of N. Thus a hash table allows one to check self-avoidance in average time $O(N)$ using only $O(N)$ words of memory. Thus we have the satisfactory property that the amount of time and space needed to check self-avoidance are both proportional to what is required just to write down the walk. References about hash tables and their implementation for self-avoiding walk problems can be found in the Notes for this chapter.

9.2 Statistical considerations

In this section we shall survey some of the statistical problems associated with Monte Carlo methods. In particular, this will lead us to the important concept of *autocorrelation times* for dynamic methods.

9.2.1 Curve-fitting and linear regression

First we shall recall some elementary statistics. If a random variable Y is normally distributed with (unknown) mean m and (known) variance σ^2, then the probability that m lies in the (random) interval $[Y - 1.96\sigma, Y + 1.96\sigma]$ is about 0.95. Thus we say $Y \pm 1.96\sigma$ is a *95% confidence interval* for m. Often the variance is also unknown, and we have to compute an estimate $\hat{\sigma}^2$ of σ^2. In this case, $Y \pm 1.96\hat{\sigma}$ is only an approximate 95% confidence interval; for the usual estimates of the variance, the 1.96 should be replaced by a suitable number from a table of the Student's t distribution (for more details, consult the statistics references in the Notes).

Now let us consider the scenario described at the beginning of this chapter, in which we attempt to estimate ν from several data points $(N_i, \hat{Y}_i)$ $(i = 1, \ldots, m)$, where N_i is chosen in advance by the experimenter and $\hat{Y}_i$ is an estimate of $\langle |\omega(N_i)|^2 \rangle$ obtained by generating a large number of random N_i-step self-avoiding walks. Let σ_i^2 be the variance of $\hat{Y}_i$; since the variance is generally not known, we will in practice need to compute an *estimate* $\hat{\sigma}_i^2$ of σ_i^2 (we shall discuss how to do this below).

To estimate ν, we begin with the scaling relation

$$\langle |\omega(N)|^2 \rangle \sim AN^{2\nu}. \tag{9.2.1}$$

We can write this asymptotic relation as an equality with (infinitely many) "correction-to-scaling" terms:

$$\langle |\omega(N)|^2 \rangle = AN^{2\nu}(1 + BN^{-\Delta} + \cdots). \tag{9.2.2}$$

The exponents of the correction terms are strictly positive, and Δ is the smallest of them, i.e. $BN^{-\Delta}$ is the dominant correction term. (Like ν, these exponents are believed to depend only on the dimension. Other forms of corrections, such as logarithms, are also possible.) Our job is to fit a curve $Y = f(N)$ to the data; to do this in a meaningful way, we must only allow a small number of parameters in the family of curves. The obvious choices are either to eliminate all of the correction-to-scaling terms, giving the two-parameter family of curves

$$Y = AN^{2\nu}, \tag{9.2.3}$$

or else to eliminate all but the dominant correction term, giving the four-parameter family

$$Y = AN^{2\nu}(1 + BN^{-\Delta}). \tag{9.2.4}$$

The form (9.2.3) is appropriate if the N_i's are all large enough so that the actual corrections to scaling are smaller than the statistical errors in the

data (i.e. smaller than σ_i). In general, however, we cannot expect this *a priori*. If we choose to work with (9.2.4), there is no guarantee that the best curve of this form will reflect the true value of Δ, since we do not know the size of the omitted correction terms (when N is small, these terms can be large, making it hard to see Δ from data corresponding to small N_i; but when N is large and the omitted terms are small, then the included term $BN^{-\Delta}$ is also small). The combination of all of the correction terms may very well show up in the data as a single "effective exponent" Δ_{eff}, which has no real relation to (9.2.2). Thus it is a very delicate business to try to estimate the true value of Δ. Rather, we may view the role of the parameter Δ in (9.2.4) as an aid to the extrapolation of a finite amount of data into the $N \to \infty$ asymptotic regime. (This represents a relatively cautious viewpoint which is definitely not universally accepted within the physics community.)

The standard statistical tool for fitting curves of the above forms to data is the method of least squares. Functions of the form (9.2.3) and (9.2.4) are examples of *regression functions*. Linear regression functions are the easiest to work with, so we begin by taking logarithms of the above two equations, obtaining

$$\log Y = \log A + 2\nu \log N \tag{9.2.5}$$

and

$$\log Y = \log A + 2\nu \log N + BN^{-\Delta}, \tag{9.2.6}$$

where the last term of (9.2.6) was obtained by the approximation $\log(1 + x) \approx x$ for x near 0. (If the reader accepts the viewpoint of the preceding paragraph that the parameter Δ should be regarded merely as an aid to extrapolation, then this approximation should cause no worries.)

Let us first focus on (9.2.5). Ordinary least squares estimation would tell us to estimate A and ν by the values that minimize the sum of squares

$$\sum_{i=1}^{m} (\log \hat{Y}_i - \log A - 2\nu \log N_i)^2.$$

This is not appropriate for us because an underlying assumption of this method is that the variance of $\log \hat{Y}_i$ is the same for every i. Instead, we should use *weighted least squares* estimation, weighting each term according to the inverse of its (estimated) variance, so that the Y_i's in which we have more confidence will have more say in determining the best fit. The general procedure is the following. Suppose that we observe independent random variables $U_1, \ldots, U_m$ where each U_i is normally distributed with mean $a + bM_i$ (where we know M_i and we want to estimate a and b) and variance v_i^2. (Our case corresponds to $b = \nu$, $a = \log A$, $U_i = \log \hat{Y}_i$ and

$M_i = 2 \log N_i$.) Then the weighted least squares estimates $\hat{a}$ and $\hat{b}$ are the values of a and b that minimize the weighted sum of squares

$$SS(a,b) \equiv \sum_{i=1}^{m} w_i (U_i - a - bM_i)^2, \tag{9.2.7}$$

where w_i is a positive "weight" (typically $1/v_i^2$, but not necessarily). The minimizing values are

$$\hat{b} = \frac{\sum w_i \sum w_i M_i U_i - \sum w_i M_i \sum w_i U_i}{\sum w_i \sum w_i M_i^2 - (\sum w_i M_i)^2} \tag{9.2.8}$$

and

$$\hat{a} = \frac{\sum w_i U_i - \hat{b} \sum w_i M_i}{\sum w_i}. \tag{9.2.9}$$

These are *unbiased estimators* of b and a (i.e. $E(\hat{b}) = b$ and $E(\hat{a}) = a$). Also, $\hat{b}$ and $\hat{a}$ are normally distributed with variances

$$\mathrm{Var}(\hat{b}) = \frac{\sum w_i}{\sum w_i \sum w_i M_i^2 - (\sum w_i M_i)^2}$$

and

$$\mathrm{Var}(\hat{a}) = \frac{\sum w_i M_i^2}{\sum w_i \sum w_i M_i^2 - (\sum w_i M_i)^2}.$$

The variances can be used to give statistical confidence intervals for a and b in the usual way. We can also formulate a test for the "goodness of fit" of our model: If the model is correct, and if the weights are given by $w_i = 1/v_i^2$, then the "residual sum of squares" $SS(\hat{a}, \hat{b})$ has a χ^2 distribution with $m - 2$ degrees of freedom.

When applying this theory in our Monte Carlo setting, we must first decide whether the estimates $\hat{Y}_i$ are normally distributed. Typically, $\hat{Y}_i$ is the average of a large number of observations

$$\hat{Y}_i = \frac{X_1 + \cdots + X_T}{T}.$$

In the case of a static method such as NRSS, the X_t's all come from different N_i-step walks, so they are i.i.d. (independent and identically distributed). The central limit theorem tells us that their average is normally distributed if T is large enough. (For an objective statistical test of normality, one can use the test of Shapiro and Wilk (1965); see Appendix A of Bratley, Fox,

and Schrage (1987).) Moreover, in the i.i.d. case, the variance of $\hat{Y}_i$ is $\mathrm{Var}(X_1)/T$, so we can estimate σ_i^2, the variance of $\hat{Y}_i$, by

$$\hat{\sigma}_i^2 = \frac{1}{T^2}\sum_{t=1}^{T}(X_t - \hat{Y}_i)^2.$$

The case of dynamic methods will be discussed in Section 9.2.2.

Suppose now that we believe that $\hat{Y}_i$ is approximately normally distributed, say with mean Y_i and variance σ_i^2. What can we say about $U_i = \log \hat{Y}_i$? Assume that σ_i is much smaller than Y_i, i.e. the uncertainty is relatively small compared to the magnitude of the quantity being estimated, as should be true in any good Monte Carlo experiment which tries to estimate something that can only be positive. Then U_i is approximately normally distributed with mean $\log Y_i$ and variance σ_i^2/Y_i^2. To see this, we write $\hat{Y}_i = Y_i + Z\sigma_i$, where Z is approximately normally distributed with mean 0 and variance 1; then

$$U_i = \log\left[Y_i\left(1 + \frac{Z\sigma_i}{Y_i}\right)\right] = \log Y_i + \log\left(1 + \frac{Z\sigma_i}{Y_i}\right) \approx \log Y_i + \frac{Z\sigma_i}{Y_i},$$

and the assertion follows. Thus $\hat{\sigma}_i^2/\hat{Y}_i^2$ is an estimate of the variance of U_i. Therefore, in the weighted least squares procedure described above, the appropriate choices of weights are $w_i = \hat{Y}_i^2/\hat{\sigma}_i^2$.

For completeness, we shall briefly describe weighted least squares estimation for more than two parameters. The framework of general linear regression is best expressed in matrix notation. Put the observed random variables $(\hat{Y}_1, \ldots, \hat{Y}_m$ in our case) into an $m \times 1$ column matrix $\mathbf{Y}$. Let β be a $p \times 1$ column vector containing the unknown parameters, and let $\mathbf{X}$ be a known $m \times p$ matrix; the model assumes that $E(\mathbf{Y}) = \mathbf{X}\beta$. (For example, in the model (9.2.6)[2]: p is 3; the entries of β are $\log A$, 2ν, and B; and the i-th row of $\mathbf{X}$ consists of the entries 1, $\log N_i$, and $N_i^{-\Delta}$.) Also let $\mathbf{V}$ be a known $m \times m$ positive definite matrix, which we assume to be the covariance matrix of $\mathbf{Y}$ (i.e. $\mathbf{V} = E[(\mathbf{Y} - \mathbf{X}\beta)(\mathbf{Y} - \mathbf{X}\beta)^T]$, where the T denotes transpose). The weighted least squares estimator is the vector $\hat{\beta}$ which minimizes

$$SS(\beta) = (\mathbf{Y} - \mathbf{X}\beta)^T\mathbf{V}^{-1}(\mathbf{Y} - \mathbf{X}\beta);$$

it is given by

$$\hat{\beta} = (\mathbf{X}^T\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{V}^{-1}\mathbf{Y}.$$

[2] Observe that in the context of *linear* regression, we must assume a fixed value for Δ in this model. The most common choice is $\Delta = 1$; sometimes renormalization group calculations suggest other values, such as $\Delta = 1/2$.

Then $\hat{\beta}$ has a multidimensional normal distribution whose mean vector is the true β and whose covariance matrix is $(\mathbf{X}^T\mathbf{V}^{-1}\mathbf{X})^{-1}$. If the model is correct, then $SS(\hat{\beta})$ has a χ^2 distribution with $m-p$ degrees of freedom.

9.2.2 Autocorrelation times: statistical theory

When a dynamic Monte Carlo experiment is performed, the observations do not form an independent sequence, and so elementary statistical methods are often not applicable. In this section we shall address the problem of how to estimate the variance of the average of a large number of observations from a dynamic Monte Carlo experiment. Once we know how to perform such estimates, we can apply the regression theory outlined in Section 9.2.1.

To be specific, suppose that $\{\omega^{[t]} : t = 1, 2, \ldots\}$ is a stationary Markov chain. (A stochastic process is said to be stationary if, for every $k \geq 0$, the joint distribution of $(\omega^{[t]}, \ldots, \omega^{[t+k]})$ is the same for every t.) For a stationary Markov chain, the distribution of $\omega^{[t]}$ for any fixed time t must be the equilibrium distribution. Every positive recurrent Markov chain is asymptotically stationary; in practice, we can assume that a Markov chain is stationary if we discard enough initial observations so that the chain has had enough time to forget any influence of its initial state and has "reached equilibrium".

Let g be a real-valued function on the state space (e.g. $g(\omega) = |\omega(N)|^2$ if the state space is $\mathcal{S}_N$). Such a function is often called an "observable" in the physics literature. Let $\langle\cdot\rangle$ denote expected value with respect to the equilibrium distribution of the chain. Then we would like to estimate $\langle g \rangle$ by the estimator

$$\hat{Y} \equiv \hat{Y}[T] \equiv \frac{1}{T}[g(\omega^{[1]}) + \cdots + g(\omega^{[T]})].$$

Since the distribution of $\omega^{[t]}$ is the stationary distribution, it is easy to see that $\hat{Y}$ is an unbiased estimator of Y. But what is its variance? This question is addressed in the following lemma.

Lemma 9.2.1 *Suppose $\{X^{[t]}\}$ is a real-valued stationary process with finite second moment. Let*

$$\hat{Y}[T] = \frac{1}{T}(X^{[1]} + \cdots + X^{[T]}).$$

For each integer k, let $C_X(k)$ denote the covariance of $X^{[t]}$ and $X^{[t+k]}$ (observe that this is independent of t by stationarity; we are implicitly re-

stricting consideration to $t \geq 1$ and $t + k \geq 1$). Let

$$v = \sum_{k=-\infty}^{\infty} C_X(k),$$

and assume that this sum converges absolutely. Then

$$\lim_{T \to \infty} T \operatorname{Var}(\hat{Y}[T]) = v.$$

Proof. We have

$$\operatorname{Var}(\hat{Y}[T]) = \frac{1}{T^2} \sum_{s,t=1}^{T} \operatorname{Cov}(X^{[s]}, X^{[t]}) = \frac{1}{T^2} \sum_{k=-(T-1)}^{T-1} (T - |k|) C_X(k).$$

The result now follows from the dominated convergence theorem. □

In the notation of the above lemma, $C_X(0)$ is the variance of $X^{[t]}$, and $C_X(k)/C_X(0)$ is called the *autocorrelation function.* The ratio $v/[2C_X(0)]$ is called the *integrated autocorrelation time*, and is denoted $\tau_{int,X}$. When the $X^{[t]}$ are independent, $\tau_{int,X} = 1/2$.

Returning to our dynamic Monte Carlo algorithm, we shall take $X^{[t]} = g(\omega^{[t]})$ in the above lemma. We now write $C_g(k)$ for the covariance of $g(\omega^{[t]})$ and $g(\omega^{[t+k]})$, and the integrated autocorrelation time is

$$\tau_{int,g} = \sum_{k=-\infty}^{\infty} \frac{C_g(k)}{2C_g(0)} = \frac{1}{2} + \sum_{k=1}^{\infty} \frac{C_g(k)}{C_g(0)}. \tag{9.2.10}$$

The lemma tells us that

$$\operatorname{Var}(\hat{Y}[T]) \sim \frac{2}{T} \tau_{int,g} \operatorname{Var}(g(\omega^{[1]})) \quad \text{as } T \to \infty. \tag{9.2.11}$$

This asymptotic relation has a very useful intuitive interpretation. If the $\omega^{[t]}$'s (and hence the $X^{[t]}$'s) were *independent*, then the variance of the average $\hat{Y}[T]$ would be given by (9.2.11) with $2\tau_{int,g}$ replaced by 1. This means that if we are using a dynamic Monte Carlo method and we want to get an estimator with the same variance as one that samples T *independent* observations, then we need $2\tau_{int,g}T$ consecutive observations from the Markov chain. In other words, $2\tau_{int,g}$ is the number of observations from the chain that we need to get one "effectively independent" data point.

So far we have neglected the question of whether or not the series defining v in Lemma 9.2.1 converges absolutely. Fortunately, the answer is that it usually does; in fact, the terms $C_X(k)$ frequently decay exponentially.

The inverse of this decay rate is known as the *exponential autocorrelation time*. Specifically, given a real-valued function g on the state space of our stationary Markov chain $\{\omega^{[t]}\}$, we define its exponential autocorrelation time to be

$$\tau_{exp,g} \equiv \limsup_{k \to \infty} \frac{k}{-\log |C_g(k)|}; \tag{9.2.12}$$

thus, the covariances $C_g(k)$ decay roughly like $\exp(-k/\tau_{exp,g})$. We also define the exponential autocorrelation time of the Markov chain to be

$$\tau_{exp} = \sup_g \tau_{exp,g}, \tag{9.2.13}$$

where the sup is over all g such that $E(g(\omega^{[t]})^2)$ is finite. This means that τ_{exp} is the relaxation time of the slowest mode in the system. As we shall see in Section 9.2.3, τ_{exp} plays an important role in measuring the rate of convergence to equilibrium from an arbitrary initial distribution. The exponential autocorrelation time could be infinite (as in the BFACF algorithm of Section 9.6.1), but this is typically not the case. In particular, as we shall see in Section 9.2.3, τ_{exp} is finite whenever the state space is finite.

Given this theoretical description of the situation, we still need to find good statistical techniques for estimating the variance of $\hat{Y}[T]$, or, equivalently, for estimating $C_g(0)\tau_{int,g}$. This kind of problem has been the focus of much research in the field of time series, and we shall limit ourselves here to a very brief discussion; the Notes at the end of the chapter give some references for additional information.

One of the simplest procedures is the method of *batched means*. Given a long sequence of observations $X^{[1]}, \ldots, X^{[T]}$ of a stationary process, divide them into some relatively small number n of equal length subsequences, or "batches". Let $b = T/n$ be the number of observations in each batch, and let Y_i be the average of the i-th batch:

$$Y_i = \sum_{j=(i-1)b+1}^{ib} X^{[j]}/b.$$

If we assume that b is much larger than τ_{exp}, then the Y_i's are approximately independent and approximately normal, each with mean $E(X^{[1]})$ and variance $\approx v/b$ where v is defined as in Lemma 9.2.1 [see for example Theorem 20.1 of Billingsley (1968) or Corollary 1.5 of Kipnis and Varadhan (1986)]. Thus the overall average $\hat{Y}[T]$ is the average of the Y_i's, and we can estimate its variance using the sample variance of the Y_i's. For a "quick and dirty" method, this one is not bad. One serious drawback of course is

the assumption that $b >> \tau_{exp}$: in particular, the results of the procedure cannot be used as a check on the assumption after the fact.

A more developed approach is the spectral analysis of time series. Briefly, this tries to estimate the infinite sum v $(= 2C_g(0)\tau_{int,X})$ by estimating each term in the infinite series. By analogy with the usual estimator for covariance, we define the following estimator of $C_X(k)$:

$$\hat{C}_X(k) = \frac{1}{T-k} \sum_{j=1}^{T-k} (X^{[j]} - \hat{Y}[T])(X^{[j+k]} - \hat{Y}[T])$$

for $k = 0, 1, \ldots, T-1$. This is a biased estimator of $C_X(k)$, but it converges to $C_X(k)$ with probability one as $T \to \infty$ by the ergodic theorem. We next define the estimators of v

$$V_{T,m} = \hat{C}_X(0) + 2 \sum_{k=1}^{m} \hat{C}_X(k)$$

(the number m is chosen by the user). We don't insist on taking $m = T - 1$ because we believe that $C_X(k)$ is close to 0 when k is large, and so $\hat{C}_X(k)$ is mostly noise when k is large. (Heuristically: $\mathrm{Var}(\hat{C}_X(k)) = O(1/T)$, so $\mathrm{Var}(V_{T,T-1}) = O(1)$ — i.e. the uncertainty does not disappear as $T \to \infty$!) How should m be chosen? One reasonable way is the following "automatic windowing" procedure: Let m be the smallest integer such that $m \geq 10V_{T,m}$. The factor 10 here is somewhat arbitrary, but the idea is that we want to make sure that we include contributions from terms that are up to several τ_{int}'s apart.

There is one more statistical issue that we must mention here, and that is the problem of *initialization bias*. In this section we have assumed that our observations come from a stationary process. Although Markov chains are *asymptotically* stationary, a simulation typically starts from a state which is *not* chosen according the equilibrium distribution. For example, in the case of self-avoiding walks, one might wish to start with a walk that is a straight line segment (for programming convenience). Thus, a simulation typically begins with an initial period which is "far from equilibrium", and it eventually "approaches equilibrium". The initial period must be removed from the data lest it introduce a bias to our estimates. Thus the experimenter must decide when the process has "reached equilibrium". The simplest procedure is to watch some observables over time until they all appear to have stabilized. There are also various statistical procedures that have been developed for removing initialization bias; see Bratley, Fox, and Schrage (1987) for a survey and references.

For concreteness, let us briefly consider the specific problem of choosing an initial state for a simulation of a Markov chain on the state space $\mathcal{S}_N$

of N-step self-avoiding walks. We could generate an initial walk using a static algorithm such as NRSS; although this would be slow, it has the theoretical advantage that we would then be starting the chain in equilibrium (exactly!), and so we would not have to worry about initialization bias at all. However, even for N around 200, it would be much faster to start with a straight walk and run until equilibrium is reached than it would be to generate a single walk by NRSS. But there are better static methods than NRSS; in particular, it is feasible to use dimerization (Section 9.3.2) to generate a single self-avoiding walk of two or three thousand steps in two dimensions (and even longer in three dimensions) to use as an initial state. (We remark that self-avoiding walks are one of the few interesting systems where there exists a feasible procedure for generating an initial state from the *exact* equilibrium distribution; nothing comparable is known for Ising-type models.)

9.2.3 Autocorrelation times: spectral theory and rigorous bounds

Consider an irreducible Markov chain with state space S, transition probabilities P, and equilibrium distribution π. Define the inner product of two complex-valued functions f and g on S to be

$$(f, g) \equiv \sum_{i \in S} \overline{f(i)} g(i) \pi(i); \tag{9.2.14}$$

the associated norm is

$$||f||_2 \equiv (f, f)^{1/2} = \left(\sum_{i \in S} |f(i)|^2 \pi(i) \right)^{1/2} \tag{9.2.15}$$

Let $l^2(\pi)$ denote the Hilbert space of the complex-valued functions f with $||f||_2$ finite. As usual, the norm of an operator T on $l^2(\pi)$ is given by

$$||T|| \equiv \sup\{||Tf||_2 : f \in l^2(\pi), ||f||_2 \leq 1\}.$$

We view P as an operator on $l^2(\pi)$ by defining

$$(Pf)(i) = \sum_{j \in S} P(i, j) f(j).$$

The operator P is a contraction on $l^2(\pi)$, i.e. $||P|| \leq 1$. To prove this, we observe that $[(Pf)(i)]^2 \leq (P(f^2))(i)$ for every i by the Schwarz inequality, and therefore

$$||Pf||_2^2 \leq \sum_{i \in S} (P(f^2)(i)) \pi(i) = ||f||_2^2 \tag{9.2.16}$$

[using (9.1.4) to get the equality].

Since $||P|| \leq 1$, all of the eigenvalues of P lie on or inside the unit circle. Moreover, using Perron-Frobenius theory one can show the following [Šidák (1964)]: since the chain is irreducible, 1 is a simple eigenvalue of P, with the constant function $\mathbf{1}$ as an eigenfunction; and 1 is the only eigenvalue of P on the unit circle if and only if the chain is aperiodic.

Define the operator Π which maps $l^2(\pi)$ to the constant functions as follows:

$$(\Pi f)(i) = \sum_{j \in S} \pi(j) f(j) \quad \text{for every } i;$$

thus $(\Pi f)(i)$ equals the expectation of f with respect to π. The basic convergence theory of Markov chains tells us that P^k converges to Π in a sense that will be made precise below. Observe that $\Pi^2 = \Pi$ and Π is self-adjoint [i.e. $(f, \Pi g) = (\Pi f, g)$], so Π is the orthogonal projection onto the space of constant functions. Also, $\Pi P = \Pi = P\Pi$ [by (9.1.4)], and so

$$(I - \Pi)P = P - \Pi = P(I - \Pi). \tag{9.2.17}$$

We shall focus on the operator $P - \Pi$, which is 0 on the subspace of constant functions and equals P on the orthogonal complement of that subspace.

For the rest of this section, we shall also assume that the Markov chain is *reversible* with respect to π, i.e. that (9.1.5) holds. This implies that P is *self-adjoint* on $l^2(\pi)$:

$$\begin{aligned} (f, Pg) &= \sum_i \overline{f(i)} \sum_j P(i,j) g(j) \pi(i) \\ &= \sum_j \sum_i \overline{f(i)} g(j) P(j,i) \pi(j) = (Pf, g). \end{aligned}$$

Since a self-adjoint operator must have real spectrum, it follows from the fact that $||P|| \leq 1$ that the spectrum of P is a subset of the interval $[-1, 1]$.

We shall now state a few facts from functional analysis. Let T be a bounded operator on a Hilbert space, and let $\sigma(T)$ be the spectrum of T. The *spectral radius* of T, denoted $r(T)$, is defined to be

$$r(T) \equiv \sup\{|\lambda| : \lambda \in \sigma(T)\};$$

it satisfies the well known "spectral radius formula"

$$r(T) = \lim_{n \to \infty} ||T^n||^{1/n} = \inf_{n \geq 1} ||T^n||^{1/n}.$$

Suppose now that T is also *self-adjoint*, so that $\sigma(T)$ is real. Then we in fact have

$$r(T) = ||T|| = ||T^n||^{1/n} \quad \text{for every } n \geq 1. \tag{9.2.18}$$

The first equality is well-known [e.g. Theorem VI.6 of Reed and Simon (1972)]; the second equality follows from $r(T) \leq ||T^n||^{1/n}$ (from the spectral radius formula), the inequality $||T^n|| \leq ||T||^n$, and the first equality. We also know

$$\begin{aligned} \inf \sigma(T) &= \inf\{(f, Tf) : ||f||_2 \leq 1\} \quad \text{and} \\ \sup \sigma(T) &= \sup\{(f, Tf) : ||f||_2 \leq 1\} \end{aligned} \tag{9.2.19}$$

(Yosida (1980), p.320); in particular, this implies the "Rayleigh-Ritz principle"

$$r(T) = \sup\{|(f, Tf)| : ||f||_2 \leq 1\}. \tag{9.2.20}$$

Finally, we have the relation

$$r(T) = \sup_f \limsup_{n \to \infty} |(f, T^n f)|^{1/n} \tag{9.2.21}$$

which we shall prove in Section 9.7.1.

Now let us return to our Markov chain. Using the notation of Section 9.2.2, we find for the stationary Markov chain $\{\omega^{[t]}\}$ that

$$\begin{aligned} C_g(k) &= E[(g(\omega^{[t]}) - \langle g \rangle)(g(\omega^{[t+k]}) - \langle g \rangle)] \\ &= \sum_i \pi(i)\,[(g - \Pi g)(i)] \left[\sum_j P^k(i,j)(g - \Pi g)(j) \right] \\ &= ((I - \Pi)g, P^k(I - \Pi)g) \\ &= \begin{cases} (g, (P - \Pi)^k g) & \text{for } k \geq 1 \\ (g, (I - \Pi)g) & \text{for } k = 0, \end{cases} \end{aligned} \tag{9.2.22}$$

where we have used $I - \Pi = (I - \Pi)^2$ and (9.2.17) in the last step. By definition, $\limsup_{k \to \infty} |C_g(k)|^{1/k} = \exp(-1/\tau_{exp,g})$ and $\tau_{exp} = \sup_g \tau_{exp,g}$, so (9.2.22) and (9.2.21) imply that $\exp(-1/\tau_{exp}) = r(P - \Pi)$; equivalently,

$$\tau_{exp} = \frac{1}{-\log r(P - \Pi)}. \tag{9.2.23}$$

Since $P - \Pi$ is self-adjoint,

$$r(P - \Pi) = ||P - \Pi|| = ||(P - \Pi)^k||^{1/k} = ||P^k - \Pi||^{1/k}. \tag{9.2.24}$$

This implies that τ_{exp} also measures the exponential rate of convergence to equilibrium when the Markov chain is not started in equilibrium. In detail, consider the metric for probability measures on S defined by

$$\rho(\phi, \psi) = \sup\{|\sum_j f(j)\phi(j) - \sum_j f(j)\psi(j)| : ||f||_2 \leq 1\}$$

[recall that $\|\cdot\|_2$ is the $l^2(\pi)$ norm of (9.2.15)]. If a Markov chain begins with the initial probability distribution ϕ at time 0, then at time k its distribution is given by the measure $(\phi P^k)(j) \equiv \sum_i \phi(i)P^k(i,j)$. For any f in $l^2(\pi)$, we have

$$\begin{aligned} &|\sum_j f(j)(\phi P^k)(j) - \sum_j f(j)\pi(j)| \\ &= |\sum_{i,j}[\phi(i) - \pi(i)][P^k(i,j) - \pi(j)]f(j)| \\ &\leq \sup\{\sum_i [\phi(i) - \pi(i)]h(i) : \|h\|_2 \leq \|P^k - \Pi\|\,\|f\|_2\}, \end{aligned}$$

and hence

$$\rho(\phi P^k, \pi) \leq \|P^k - \Pi\|\rho(\phi, \pi) = \exp(-k/\tau_{exp})\rho(\phi, \pi). \tag{9.2.25}$$

Equation (9.2.25) has the following practical interpretation: it tells us that if we begin from an initial distribution which is different from π and run the Markov chain for $10\tau_{exp}$ iterations, say, then the deviation from equilibrium (with respect to the metric ρ) is at most e^{-10} (about 0.00004) times the initial deviation. On the one hand, it is usually very difficult to get information about the size of τ_{exp} (either rigorously or numerically), so this is rarely a practical criterion for ensuring that the simulation has "reached equilibrium". On the other hand, the convergence to equilibrium could in fact be much faster than the upper bound of (9.2.25) indicates, so not knowing τ_{exp} may not be a real disadvantage. Ultimately, one has to analyze the data to determine empirically when the process is sufficiently close to equilibrium (see the discussion at the end of Section 9.2.2).

We remark that when the state space S is finite, then the spectrum of $P - \Pi$ is a finite subset of $(-1, 1)$ (assuming aperiodicity), and in particular τ_{exp} must be finite.

Up to now, we have been talking about the spectral radius of $P - \Pi$, but in Monte Carlo work one is usually just interested in the spectrum near $+1$ rather than near -1. An eigenvalue at -1 causes τ_{exp} to be infinite, but for a trivial reason: it happens if and only if the Markov chain is periodic with an even period, and so $\rho(\phi P^k, \pi)$ typically does not even converge to 0 because the chain always remembers which part of the state space it started in. But this does not prevent the *averages* $\hat{Y}[T]$ from converging rapidly to the correct values. So let us define the *modified autocorrelation time*

$$\tau'_{exp} = \frac{1}{-\log[\sup \sigma(P - \Pi)]}. \tag{9.2.26}$$

Then it will be shown in Section 9.7.1 that for every g in $l^2(\pi)$

$$\tau_{int,g} \leq \frac{1}{2}\left(\frac{1+\exp[-1/\tau'_{exp}]}{1-\exp[-1/\tau'_{exp}]}\right) = \tau'_{exp}\left[1+O(1/\tau'_{exp})\right] \qquad (9.2.27)$$

for τ'_{exp} bounded away from 0.

The following result will be proven in Section 9.7.1. Its corollary below will be used a number of times in this chapter (see Sections 9.4.1, 9.5.1, and 9.6.1). We remind the reader that the covariances $C_g(k)$ and the various autocorrelation times are always defined in terms of the *stationary* Markov chain corresponding to P and π.

Proposition 9.2.2 *Suppose that P is reversible with respect to π. Then for any nonconstant g in $l^2(\pi)$,*

$$\tau_{int,g} \geq \frac{1}{2}\left(\frac{1+\rho_g(1)}{1-\rho_g(1)}\right) = \frac{1}{1-\rho_g(1)} - \frac{1}{2},$$

where

$$\rho_g(1) = \frac{C_g(1)}{C_g(0)}.$$

Corollary 9.2.3 *Suppose that P is reversible with respect to π, and let g be a function in $l^2(\pi)$. Assume that there is a finite constant A such that $|g(i)-g(j)| < A$ whenever $P(i,j) > 0$ (i.e. the value of g can never change by more than A during a single step of the Markov chain). Then*

$$\tau_{int,g} \geq \frac{2C_g(0)}{A^2} - \frac{1}{2}.$$

Proof. First we list the following identities, which may be verified by direct calculation:

$$\begin{aligned} C_g(0)(1-\rho_g(1)) &= C_g(0) - C_g(1) \\ &= (g,(I-P)g) \\ &= \frac{1}{2}\sum_{i,j}\pi(i)P(i,j)|g(i)-g(j)|^2. \end{aligned} \qquad (9.2.28)$$

From (9.2.28), we see that $C_g(0)(1-\rho_g(1)) \leq A^2/2$. The result now follows immediately from Proposition 9.2.2. □

As a further application of the identities (9.2.28), we have the following result:

Proposition 9.2.4 *Suppose that P_1 and P_2 are transition probabilities of two Markov chains which are reversible with respect to the same π, and assume that $P_1(i,j) \geq P_2(i,j)$ whenever $i \neq j$. Then their respective modified autocorrelation times satisfy $\tau'_{exp}(P_1) \leq \tau'_{exp}(P_2)$.*

Proof. For $k = 1, 2$, we see from (9.2.19) that

$$\sup \sigma(P_k - \Pi) = \sup\{(f, (I - \Pi)f) - (f, (I - P_k)f) : \|f\|_2 \leq 1\}.$$

In view of (9.2.28), this implies that $\sup \sigma(P_1 - \Pi) \leq \sup \sigma(P_2 - \Pi)$. The proposition then follows from (9.2.26). □

Remark. Caracciolo, Pelissetto, and Sokal (1990) prove several generalizations of Proposition 9.2.4, including the result due to Peskun (1973) that the same hypotheses imply that $\tau_{int,f}(P_1) \leq \tau_{int,f}(P_2)$ for every f. The intuition behind these results is clear: since P_1 makes more transitions than P_2, it approaches equilibrium faster.

9.3 Static methods

In this section we shall discuss a number of static Monte Carlo algorithms. These algorithms generate either a sequence of independent self-avoiding walks or a sequence of independent batches of self-avoiding walks (the walks within each batch possibly being highly correlated).

9.3.1 Early methods: strides and biased sampling

Two methods of generating independent sequences were discussed in Section 9.1, namely Elementary Simple Sampling and Non-Reversed Simple Sampling; both were seen to require an exponentially large amount of computer time for each self-avoiding walk generated. A natural generalization of these methods uses "strides" to build walks instead of single steps. An *m-step stride* is a self-avoiding walk of length m. For the following algorithm, let m be a fixed nonnegative integer.

m-Step Stride Method (SM(m)). This algorithm generates a self-avoiding walk of length km (k an integer). It requires a list $\psi[1], \ldots, \psi[c_m]$ of all m-step self-avoiding walks.

1. Set W to be the 0-step walk consisting of the single site at the origin. Set $i = 0$.
2. Increase i by one. Choose an integer J uniformly at random from $\{1, \ldots, c_m\}$. Redefine W to be $W \circ \psi[J]$, the concatenation of $\psi[J]$ to the current W.

3. If W is not self-avoiding, then go back to Step 1. Otherwise, go to Step 2 if $i < k$, and stop if $i = k$.

The average amount of computer time required to generate one N-step self-avoiding walk using this algorithm is

$$T_{SM(m)} = \frac{(c_m)^{N/m}}{c_N} = \left(\frac{c_m^{1/m}}{\mu}\right)^{N+o(N)}$$

This still grows exponentially in N, but at a slow rate if m is large. Of course, the larger m is, the more overhead must be invested in preparing and storing the list of all m-step walks.

One easy way to improve the Stride Method (for a given m) is in Step 2 to choose $\psi[J]$ from among only those walks whose first bond is not in the direction opposite to the last bond of the current W. A more sophisticated approach, requiring additional work in advance, is the following. For each $i = 1, \ldots, c_m$, make a list L_i containing all values of j such that $\psi[i] \circ \psi[j]$, the concatenation of $\psi[j]$ to $\psi[i]$, is self-avoiding. Then in Step 2 only choose the next J from the list L_J corresponding to the current J. Unfortunately, since the lists do not all have the same length, this will not generate walks with uniform distribution on $\mathcal{S}_{km}$ unless we exercise some caution. Specifically, we could let $L = \max_i |L_i|$ (where $|L_i|$ denotes the length of the list L_i), and replace Steps 1 and 2 above as follows.

1′. Choose $J(1)$ uniformly at random from $\{1, \ldots, c_m\}$. Set $W = \psi[J(1)]$ and set $i = 1$.

2′. Increase i by one. Choose $J(i)$ uniformly at random from $\{1, \ldots, L\}$. If $J(i) > |L_{J(i-1)}|$, then go back to Step 1′ and start over; otherwise, redefine W to be the concatenation of the $J(i)$-th walk on the list $L_{J(i-1)}$ to the current W.

Again we see the usefulness of a "rejection" step (occurring here when $J(i) > |L_{J(i-1)}|$) in producing the desired distribution. Without this, our method would suffer from the same flaw as the MSAW algorithm of Section 9.1. Having sounded these warnings, let us now say that all is not necessarily lost if we generate self-avoiding walks with a nonuniform distribution, for we can still estimate interesting quantities by reweighting our observations, as we shall now explain.

Suppose that $\omega^{[1]}, \ldots, \omega^{[m]}$ is an i.i.d. sample from $\mathcal{S}_N$ with a common known probability distribution

$$q(v) \equiv \Pr\{\omega^{[1]} = v\} \qquad (v \in \mathcal{S}_N)$$

which is not uniform but is strictly positive for every v (for example they could be generated by the MSAW algorithm). Suppose that we wish to estimate some quantity $\langle f(\omega)\rangle_N$, where f is a real-valued function on $\mathcal{S}_N$ and $\langle\cdot\rangle_N$ denotes the expectation with respect to the uniform distribution of ω on $\mathcal{S}_N$. If we define the reweighted average

$$Y_m^f \equiv \frac{1}{m}\sum_{i=1}^{m}\frac{f(\omega^{[i]})}{q(\omega^{[i]})},$$

then the expectation of Y_m^f is $c_N\langle f(\omega)\rangle_N$ (that is, Y_m^f/c_N is an *unbiased estimator* of $\langle f(\omega)\rangle_N$). This is because

$$E\left(\frac{f(\omega^{[i]})}{q(\omega^{[i]})}\right) = \sum_{v\in\mathcal{S}_N}\left(\frac{f(v)}{q(v)}\right)q(v) = c_N\left(\frac{1}{c_N}\sum_{v\in\mathcal{S}_N}f(v)\right). \qquad (9.3.1)$$

In particular, if we take f identically 1, then Y_m^1 is an unbiased estimator of c_N. Since the $\omega^{[i]}$'s are i.i.d., the strong law of large numbers guarantees that Y_m^f converges to $c_N\langle f(\omega)\rangle_N$ as $m\to\infty$, with probability one. Therefore, if we define the ratio

$$R_m^f \equiv Y_m^f/Y_m^1,$$

then R_m^f converges to $\langle f(\omega)\rangle_N$ as $m\to\infty$, with probability 1.

This theory can be applied to the case of walks generated by the MSAW algorithm, once we compute the function q. This was done for two examples of four-step walks in the paragraph following the statement of the algorithm in Section 9.1. In general, suppose that $v = (v(0),\ldots,v(N))$ is a self-avoiding walk. For each $i = 0,\ldots,N-1$, let t_i be the number of neighbours of $v(i)$ that are not in the set $\{v(0),\ldots,v(i-1)\}$. Then $q(v)$ is the product of the reciprocals of $t_0,\ldots,t_{N-1}$.

This method is often referred to as "inversely restricted sampling" or "biased sampling"; it is closely related to "importance sampling" [see for example Hammersley and Handscomb (1964) or Bratley, Fox and Schrage (1987)]. It was originally used by Rosenbluth and Rosenbluth (1955) for the function $f(\omega) = |\omega(N)|^2$. Earlier, Hammersley and Morton (1954) had used a slight variant of Y_m^1 to estimate c_N.

Biased sampling has some apparent drawbacks:

- Long walks will eventually become "trapped"; this could lead to many attempts being necessary to generate a single walk, unless we had a mechanism of avoiding steps that would lead into a trap. (We remark that a Monte Carlo study by Hemmer and Hemmer (1984) concluded that walks in $\mathbf{Z}^2$ survive for 71 steps before being trapped, on average.)

- The estimator R_m^f is not unbiased in general. However, McCrackin (1972) showed that $E(R_m^f) - \langle f(\omega)\rangle_N$ is of order m^{-1}, and hence for large m the difference is negligible compared with the ubiquitous $m^{-1/2}$ statistical error inherent in i.i.d. sampling schemes.
- The weights $1/q$ vary considerably, and a typical experiment is likely to end up with most of the overall weight coming from a very small fraction of the observations [Hammersley and Handscomb (1964), Batoulis and Kremer (1988)]. That is, the variance of the estimator R_m^f is likely to be uncomfortably large for any practical value of m. One might try to improve this situation by a variant of importance sampling, in which the possibilities in Step 2 of the MSAW algorithm are weighted so that the walk is encouraged to spread out faster (the original MSAW produces walks that tend to be more compact than typical self-avoiding walks). However, any such reweighting method where the distribution being sampled is substantially different from the desired (uniform) distribution could quite easily encounter the same problems, and the situation is rather delicate. Some work in this direction is surveyed in Kremer and Binder (1988, Sec. 2.1.2).

9.3.2 Dimerization

A different method of generating self-avoiding walks uniformly on $\mathcal{S}_N$ is *dimerization*, which is essentially a recursive procedure. The idea is that if we wish to generate an N-step self-avoiding walk, then we generate two independent $(N/2)$-step self-avoiding walks ("dimers") and try to concatenate them. If the result is self-avoiding, we are done; otherwise, we discard both dimers and start again. To generate each of the $(N/2)$-step walks, we generate two $(N/4)$-step walks and try to concatenate them, and so on. The recursion can stop at the k-th level if there is a fast way to generate self-avoiding walks of length $N/2^k$. For example, 10-step walks are easy to generate by Non-Reversed Simple Sampling, so only three levels are needed to create an 80-step walk by dimerization. We can express this as the following recursive procedure.

DIM(N). This procedure generates one N-step self-avoiding walk ω uniformly from $\mathcal{S}_N$. Here N_0 is a fixed small integer (e.g. $N_0 = 10$).

1. If $N \le N_0$, then generate an N-step walk ω by NRSS and then stop.
2. ($N > N_0$) Set $N_1 = \lfloor N/2 \rfloor$ and $N_2 = N - N_1$.
3. Recursively perform DIM(N_1) and DIM(N_2), yielding the self-avoiding walks ω^1 and ω^2 respectively.

4. Set $\omega = \omega^1 \circ \omega^2$, the concatenation of ω^2 to ω^1. If ω is self-avoiding, then stop; otherwise, return to Step 2 and start over.

We remark that NRSS in Step 1 could be replaced by any other method that generates self-avoiding walks uniformly.

We shall use the following lemma to see that the end product ω is in fact uniformly distributed, as well as to investigate the efficiency of dimerization.

Lemma 9.3.1 *Let M and N be positive integers. Let $v^1, v^2, \ldots$ be independent self-avoiding walks uniformly distributed on $\mathcal{S}_M$, and let $\varphi^1, \varphi^2, \ldots$ be independent self-avoiding walks uniformly distributed on $\mathcal{S}_N$. For each $i \geq 1$, let ψ^i denote the concatenation of φ^i to v^i. Let τ be the smallest i such that ψ^i is self-avoiding. Then ψ^τ is uniformly distributed on $\mathcal{S}_{M+N}$, and*

$$E(\tau) = \frac{c_M c_N}{c_{N+M}}. \tag{9.3.2}$$

Proof. For any fixed i we have

$$\Pr\{\psi^i \text{ is self-avoiding}\} = \frac{c_{M+N}}{c_M c_N};$$

call this quantity p. Then τ has a geometric distribution, i.e.

$$\Pr\{\tau = i\} = (1-p)^{i-1}p \qquad (i \geq 1)$$

so $E(\tau) = 1/p$, which proves (9.3.2). Now let ω be any fixed $(M+N)$-step self-avoiding walk, and let ω' and ω'' be the unique M-step and N-step walks whose concatenation $\omega' \circ \omega''$ is ω. Then

$$\begin{aligned}
\Pr\{\psi^\tau = \omega\} &= \sum_{i=1}^{\infty} \Pr\{\tau = i \text{ and } \psi^i = \omega\} \\
&= \sum_{i=1}^{\infty} (1-p)^{i-1} \Pr\{v^i = \omega' \text{ and } \varphi^i = \omega''\} \\
&= \sum_{i=1}^{\infty} (1-p)^{i-1} \frac{1}{c_M} \frac{1}{c_N} \\
&= \frac{1}{c_{N+M}},
\end{aligned}$$

which proves the lemma. □

This lemma shows that in the procedure DIM(N), the final walk ω is uniformly distributed provided that the walks ω^1 and ω^2 are uniformly distributed. We know that this will be true if N_1 and N_2 are small enough

(since Step 1 is completely reliable), and so the uniformity of ω follows by induction on the number of levels in the recursion.

We now shall discuss the efficiency of dimerization, under the scaling assumption (1.1.4), i.e.

$$c_N \sim A\mu^N N^{\gamma-1}.$$

For simplicity, we assume $N = 2^k N_0$, where k is the number of levels of recursion. Let $T_{DIM(N)}$ denote the expected amount of time for the procedure DIM(N) to produce a walk. By (9.3.2), the average number of pairs of ($N/2$)-step walks that must be generated before we get a pair whose concatenation is self-avoiding is $(c_{N/2})^2/c_N$, which is asymptotic to $A(N/4)^{\gamma-1}$ by the above scaling assumption. This gives us the recursive relation

$$T_{DIM(N)} \sim BN^{\gamma-1}(2T_{DIM(N/2)})$$

(where $B = A/4^{\gamma-1}$). (We have omitted the amount of time required to check whether the two dimers intersect each other, but since this time is $O(N)$, it will be seen to be negligible compared with $2T_{DIM(N/2)}$, the time required to generate the two dimers.) Iterating this relation k times (and assuming the approximate validity of our scaling assumption all the way down to N_0) yields

$$T_{DIM(N)} \approx \frac{(2BN^{\gamma-1})^k}{2^{(\gamma-1)k(k-1)/2}} T_{DIM(N_0)} = d_0 N^{d_1 \log_2 N + d_2}, \qquad (9.3.3)$$

where the d_i are independent of N:

$$d_1 = \frac{\gamma-1}{2}, \qquad d_2 = \frac{\gamma-1}{2} + \log_2(2B) = \frac{5-3\gamma}{2} + \log_2 A,$$

and d_0 depends on N_0. We thus conclude that the growth of $T_{DIM(N)}$ is slower than exponential in N. We also notice that the anticipated values for d_1 are small: according to (1.1.11), we expect d_1 to be 11/64 in two dimensions, 0.081... in three, and 0 in four or more dimensions. In particular, since it is known rigorously that $\gamma = 1$ in five or more dimensions (see Theorem 6.1.1), the above argument can be made into a rigorous proof that $T_{DIM(N)}$ grows *polynomially* in five or more dimensions. We also note that d_2 is small in high dimensions: in $d = 5$ we have the rigorous bound $d_2 = 1 + \log_2 A \le 1 + \log_2 1.493 \le 1.58$, and it is even smaller for $d \ge 6$ (see Remark following Theorem 6.1.1).

It is tempting to try to squeeze more data out of dimerization than just the information contained in the final N-step walk. For example, to estimate ν as described at the beginning of Section 9.1, one might try to use all the generated subwalks to get estimates of $\langle|\omega(N/2^i)|^2|\rangle_{N/2^i}$ for $i = 0, \ldots, k$.

This will give an unbiased estimate for each i, but the $k+1$ estimates will be mutually correlated; this makes it difficult to find a confidence interval for ν using classical linear regression theory (Section 9.2.1). Things look better if we are trying to estimate γ. For $n = N, N/2, \ldots, N/2^{k-1}$, let $\tau_n[j]$ denote the number of attempts needed to produce the j-th n-step self-avoiding walk (i.e. the number of pairs of $(n/2)$-step walks that are concatenated after the $(j-1)$-th success until the j-th success). As discussed above,

$$E(\tau_n[j]) \sim \frac{A}{4^{\gamma-1}} n^{\gamma-1},$$

so one could try using linear regression here. If we think of repeating DIM(N) indefinitely to produce an infinite sequence of N-step self-avoiding walks, then one can easily see that all of the random variables $\tau_n[j]$ ($n = N, N/2, \ldots, N/2^{k-1}$, $j \geq 1$) are *independent.* (This is essentially because the number of attempts needed to generate an n-step walk is independent of the walk itself.)

Suppose that we wish to generate m N-step walks by dimerization and use the $\tau_n[j]$ data to estimate γ. At the top level, we get m independent observations of $\tau_N[1]$. At the next level, we get a random number of independent copies of $\tau_{N/2}[\cdot]$: in fact, this random number is exactly $2(\tau_N[1] + \cdots + \tau_N[m])$. Thus there is some dependence between the data at different levels, but one can argue that it is negligible when m is large. A more serious difficulty with this scheme is its efficiency. It produces much more data for small n than for large n (in fact, more than twice as much data for $N/2^{i+1}$ than for $N/2^i$), but this is where we have the least confidence in our scaling assumption. So it is not clear how useful this method can be for estimating γ.

9.3.3 Enrichment

The enrichment method attempts to overcome the high attrition rate of simple sampling by reusing intermediate-length walks many times. This method was originally used by Wall and Erpenbeck (1959). The basic procedure requires two integer parameters, s and t. We first attempt to generate s-step self-avoiding walks by NRSS (or a similar method). Each time that we get an s-step walk, we make t (identical) copies of it and we attempt to extend each copy independently by NRSS to length $2s$. Similarly, each time that we get a self-avoiding walk of length $2s, 3s, \ldots$, we make t copies of that walk, each of which then evolves independently. The result will be a collection of self-avoiding walks of various lengths (all multiples of s). There will be a great deal of correlation between some of these walks, because they will have exactly the same first s (or $2s$, or $3s, \ldots$) steps;

but any two walks which are not extensions of copies of the same initial s-step walk will be statistically independent. Thus the enrichment method produces several independent groups of self-avoiding walks, but the walks within each group are highly correlated. Finding the correct statistical approach to handling these correlations remains an open problem.

Let M_{ks} denote the number of ks-step walks that are produced while performing this method. Then M_s is the number of independent groups; this number in practice is likely to be fixed in advance by the experimenter (of course, M_{ks} is random for $k \geq 2$). In the subsequent analysis, we shall assume for convenience that $M_s = 1$. The probability that a single attempt to extend a ks-step walk to a $(k+1)s$-step walk succeeds is

$$\frac{c_{(k+1)s}}{(2d-1)^s c_{ks}} \sim \left(\frac{\mu}{2d-1}\right)^s .$$

We can think of $M_s, M_{2s}, \ldots$ as a *branching process* in which M_{ks} represents the number of "individuals" alive in the k-th generation, and each individual reproduces independently, the number of offspring of an individual being a binomial random variable with parameters t and $p \approx (\mu/(2d-1))^s$. No individual survives more than one generation. We can also think of every individual having t offspring, but each offspring only having probability p of reaching maturity. For more about branching processes, see for example Feller (1968) or Karlin and Taylor (1975). Strictly speaking, p is different for each generation, so we really have a time-inhomogeneous branching process. However we are not going to prove anything rigorously here, and it will be convenient to ignore this fact.

Given the number of js-step walks, the expected number of $(j+1)s$-step walks produced is

$$E(M_{(j+1)s}|M_{js}) = tpM_{js},$$

and by induction we conclude that $E(M_{ks}) = (tp)^k$. If $tp < 1$, then $E(M_{ks})$ decays exponentially: i.e. the branching process dies out exponentially fast. In this case, we do not expect to observe many long walks, and this method should not be much of an improvement over ordinary NRSS. If $tp > 1$, then there is a positive probability of a population explosion: that is, of M_{ks} increasing exponentially forever. This will lead to an enormous group of highly correlated walks. If $tp = 1$, then the branching process is "critical": it will die out eventually, but the expected time until this happens is infinite. This should produce some large walks, but there can be no population explosion of a single group. The preceding intuitive arguments are supported by the theory of branching processes. (This three-way classification is a hallmark of critical phenomena; in fact, the above branching process is essentially the same as percolation on an infinite tree in which

every site has $t+1$ neighbours.) From this discussion, we conclude that the best choice of parameters is to take t equal to $1/p$, i.e.

$$t \approx \left(\frac{2d-1}{\mu}\right)^s .$$

One can improve the enrichment method by combining it with the dimerization approach, as follows. Suppose that a self-avoiding walk ω of length ks has just been generated. Make t copies of this walk. For each copy, generate an s-step self-avoiding walk (by NRSS or some other method) completely independently of ω, and then try to concatenate it with ω. If the result has no intersections, then we have successfully produced a $(k+1)s$-step self-avoiding walk, which we can now copy t times, and so on; otherwise, the attempt fails, and this copy of ω is no longer used. The probability of a success for such an attempt is

$$\frac{c_{(k+1)s}}{c_{ks}c_s} \sim \frac{(1+\frac{1}{k})^{\gamma-1}}{As^{\gamma-1}}$$

[by the usual scaling assumption (1.1.4)]. The above discussion then suggests taking t to be the inverse of this probability. (Note that allowing t to vary with k does not bias our results, whereas allowing t to depend upon the generated walks could easily introduce significant biases.) This method appears to be significantly more efficient than ordinary enrichment, but of course it still has the problem that walks within groups are highly correlated. Variants of this method have been used by Grishman (1973) and Rapaport (1985).

A closely related method has been proposed by Redner and Reynolds (1981). Its philosophy is a bit different, in that it estimates the susceptibility and other generating functions directly. A simple version of their method may be stated as follows.

Redner-Reynolds Algorithm. This algorithm generates random sets of self-avoiding walks $A_i \subset \mathcal{S}_i$ $(i \geq 0)$. It requires a parameter z between 0 and 1. We denote the $2d$ (positive and negative) unit vectors of $\mathbf{Z}^d$ by $e_1, \ldots, e_{2d}$.

1. Let A_0 be the set consisting of the 0-step walk at the origin. Set $i = 0$. (Initially, A_k is empty for every $k \geq 1$.)
2. Independently, for each walk ω in A_i, and for each $j = 1, \ldots, 2d$: With probability $1-z$, do nothing; otherwise (i.e. with probability z) try to add a step e_j to ω, and if the result is self-avoiding, then put it in A_{i+1}.
3. Increase i by one and go back to Step 2.

The algorithm stops when some A_i is empty. This algorithm is essentially a direct exact enumeration procedure in which each possibility is only pursued with probability z. Any given N-step self-avoiding walk is generated with probability z^N and so the expected cardinality of A_N is $c_N z^N$. Thus the total number of generated walks is an unbiased estimator for the susceptibility:

$$E\left(\sum_{N=0}^{\infty} |A_N|\right) = \chi(z).$$

In particular, for the interesting case $z < z_c = \mu^{-1}$, the Redner-Reynolds algorithm terminates in finite time with probability one. One can just as easily get estimates of other quantities: for example, the sum of the squares of the end-to-end distances of all of the generated walks is an unbiased estimator of $\chi(z)\xi_2(z)^2$, where $\xi_2(z)$ is the correlation length of order two defined in (1.3.18).

9.4 Length-conserving dynamic methods

In this section we shall look at dynamic Monte Carlo methods that generate walks having a fixed number of steps N. Each method of this type corresponds to a Markov chain that takes a self-avoiding walk and tries to change it in a random way to get another self-avoiding walk of the same length. The Verdier-Stockmayer algorithm, described in Section 9.1, is an example of such a method.

The algorithms that we shall consider in this section are of the following form.

> *Generic Fixed-Length Dynamic Algorithm.* Generates a Markov chain $\{\omega^{[t]} : t = 0, 1, \ldots\}$ on the state space $\mathcal{S}_N$ which is reversible with respect to the uniform distribution on $\mathcal{S}_N$.

1. Let $\omega^{[0]}$ be any self-avoiding walk in $\mathcal{S}_N$. Set $t = 0$.
2. Use a certain randomized procedure to define a new walk $\tilde{\omega} = (\tilde{\omega}(0), \ldots, \tilde{\omega}(N))$, which is not necessarily self-avoiding.
3. If $\tilde{\omega}$ is self-avoiding, then set $\omega^{[t+1]} = \tilde{\omega}$; otherwise, set $\omega^{[t+1]} = \omega^{[t]}$.
4. Increase t by one and go to Step 2.

Usually, it will be fairly routine to check reversibility, but questions about irreducibility (ergodicity) may require some work.

Before going on, we first make some remarks about conventions for this section. We shall always use N to denote the length of the walks being generated; N is an arbitrary integer which has been fixed (by the person

running the experiment). The state space of the corresponding Markov chain is $\mathcal{S}_N$. If the algorithm changes the first part of the current walk, then its initial point may no longer be the origin (as in Step 3(c) of the V-S algorithm); in such a case, we will always implicitly assume that the resulting walk is translated so that its initial step is the origin, thereby staying in the set $\mathcal{S}_N$. (Alternatively, we can think of $\mathcal{S}_N$ as the set of equivalence classes of all N-step self-avoiding walks modulo translation; then the starting point of a generated walk is irrelevant, so there is no need to worry about translating back to the origin.) The transition probabilities will always be written $P(\cdot,\cdot)$.

9.4.1 Local algorithms

A *local* algorithm operates on walks by attempting to change only a few contiguous sites (and bonds) of the current walk at a time. The Verdier-Stockmayer algorithm is the prototype of this class of methods. Typically, a local algorithm chooses a small subwalk of the current walk at random, and attempts to replace it with a different (self-avoiding) subwalk having the same length and the same endpoints (unless the chosen subwalk includes an endpoint of the entire walk, in which case that endpoint may move). We keep the new walk if it is self-avoiding and reject it otherwise. The subwalk that we delete may uniquely determine the subwalk that replaces it (as in Step 3(a) of the V-S algorithm); alternatively, each possible subwalk may have a corresponding list of possible replacements, from which one must be chosen at random (as in Steps 3(b) and 3(c) of the V-S algorithm). Some examples are given in Figure 9.3.

The main theoretical result about these algorithms is that *none* of them is irreducible: in fact, for any given initial self-avoiding walk, the number of different walks that can be obtained from this walk by such an algorithm is exponentially smaller than c_N (for large N). Before we prove this, we shall first make our terms more precise.

Let $k \geq 1$ be a fixed integer, and let ω and ω' be N-step walks. Then we say that ω *can be transformed into* ω' *by a* k*-site move* if there exists an i ($0 \leq i \leq N-k+1$) such that $\omega(j) = \omega'(j)$ for every $j = 0, 1, \ldots, i-1, i+k, \ldots, N$—that is, if ω and ω' are the same except for at most k contiguous sites. (Observe that the initial points of ω and ω' may be different if $i = 0$; similarly for their last points if $i = N-k+1$.) We say that an algorithm is a *k-site algorithm* if the following holds: $P(\omega,\omega') > 0$ only if ω can be transformed into ω' by a k-site move. Thus the V-S algorithm is a 1-site algorithm. Finally, a length-conserving algorithm is said to be *local* if it is a k-site algorithm for some finite k. (Here, k must be independent of N; the term "algorithm" technically refers to a collection of algorithms, one

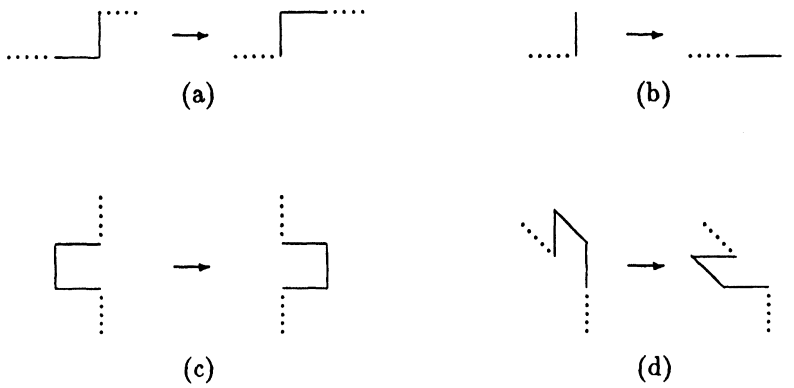

Figure 9.3: Some local length-conserving transformations. The transformation of (b) depicts the movement of an endpoint. The Verdier-Stockmayer algorithm uses (a) and (b), which are "one-site moves". Transformations (c) and (d) are "two-site moves" [(d) is shown in three dimensions].

for each N, but they are really all defined by exactly the same rules, so we use the word algorithm in the singular.)

We can define the "most general" k-site algorithm according to the recipe for the Generic Fixed-Length Dynamic Algorithm, where Step 2 is designed to allow transitions to any $\tilde{\omega}$ into which $\omega^{[t]}$ can be transformed by a k-site move. It is not hard to guarantee reversibility [Equation (9.1.6)]; for example, we can use the following rule.

2. Choose I uniformly at random from $\{0, 1, \ldots, N-k+1\}$. Set $\tilde{\omega}(l) = \omega^{[t]}(l)$ for every $l < I$ and every $l \geq I+k$. If $0 < I < N-k+1$, then randomly choose a $(k+1)$-step self-avoiding walk ω^* from among those walks in $\mathcal{S}_{k+1}$ satisfying $\omega^*(k+1) - \omega^*(0) = \omega^{[t]}(I+k) - \omega^{[t]}(I-1)$. If I is 0 or $N-k+1$, then randomly choose a k-step walk ω^* from $\mathcal{S}_k$. Then $\tilde{\omega}(I), \ldots, \tilde{\omega}(I+k-1)$ are obtained by translating ω^* so that it begins at $\omega^{[t]}(I-1)$ (or, if $I = 0$, so that it ends at $\omega^{[t]}(k)$).

Then for any two distinct N-step self-avoiding walks ω and ω',

$$P(\omega, \omega') = \frac{1}{N-k+2} \left[\sum_{i=1}^{N-k} F_i(\omega, \omega') \frac{1}{c_{k+1}(\omega(i-1), \omega(i+k))} \right.$$

$$+(F_0(\omega,\omega') + F_{N-k+1}(\omega,\omega'))\frac{1}{c_k}\Big]$$
$$= P(\omega',\omega),$$

where $F_i(\omega,\omega')$ is 1 if $\omega(l) = \omega'(l)$ for every $l < i$ and every $l \geq i+k$, and it is 0 otherwise. Thus we see that P is symmetric, and moreover that $P(\omega,\omega') > 0$ if and only if ω can be transformed into ω' by a k-site move. We shall call this algorithm the *Maximal k-Site Algorithm* (MAX(k)).

Observe that two N-step walks ω and υ are in the same ergodicity class of MAX(k) if and only if there exists a finite sequence of N-step walks $\omega \equiv \omega^{(0)},\omega^{(1)},\ldots,\omega^{(m)} \equiv \upsilon$ such that $\omega^{(i)}$ can be transformed into $\omega^{(i+1)}$ by a k-site move for every $i = 0,\ldots,m-1$. In particular, any ergodicity class of any other k-site algorithm is contained in an ergodicity class of MAX(k).

It is not hard to see that the Verdier-Stockmayer algorithm is not irreducible in general. In $\mathbf{Z}^2$, the 17-step walk $\mathrm{ENW^2S^2E^5N^2W^2SE}$ (Figure 9.2 in Section 9.1) cannot be transformed into any other self-avoiding walk by a 1-site move. We say that this walk is *frozen* (with respect to 1-site algorithms). In $\mathbf{Z}^3$, the V-S algorithm is not irreducible because of knot-like configurations: Figure 9.4 shows a 20-step walk which is in a different ergodicity class from, say, the straight walk for any 1-site or 2-site algorithm.

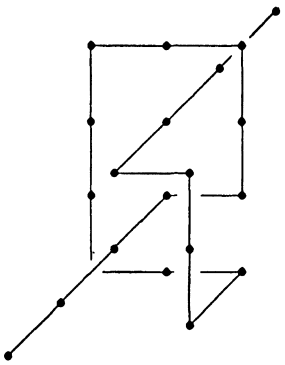

Figure 9.4: A knot-like walk in $\mathbf{Z}^3$ which cannot be transformed into a straight walk using 1-site or 2-site moves.

The observation that 1-site algorithms have frozen configurations in $\mathbf{Z}^2$ is generalized in the next theorem.

Theorem 9.4.1 *Let $d = 2$. For any integer $k \geq 1$ and any $r \geq k$, there exists a $(6r + 17)$-step self-avoiding walk which cannot be transformed into any other $(6r + 17)$-step walk by k-site moves.*

The idea of the proof is the following construction. Let $\psi^{(r)}$ be the $(6r+17)$-step walk

$$N^r E S^{r+1} W^2 N^{r+2} E^5 S^{r+2} W^2 N^{r+1} E S^r$$

(see Figure 9.5). If $r \geq k$, then $\psi^{(r)}$ is frozen under k-step moves. The details of the proof are given in Section 9.7.2. We remark that the conclu-

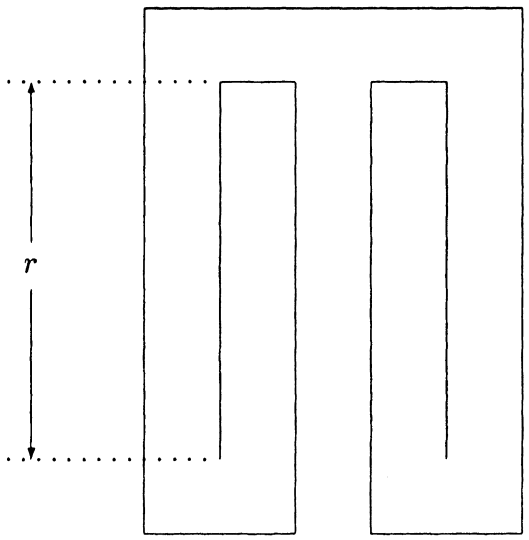

Figure 9.5: The walk $\psi^{(r)}$ from the proof of Theorem 9.4.1.

sions of this theorem are not restricted to lengths of the form $N = 6r + 17$. In fact, for every k it is true that for all sufficiently large N there exists an N-step self-avoiding walk which is frozen with respect to k-site algorithms [Madras and Sokal (1987)].

The next theorem discusses the cardinality of the largest ergodicity class (or CLEC for short) of local algorithms. It proves that for $d = 2$ or 3 the CLEC is exponentially smaller than the cardinality of the entire state space. Thus, even if we ran a Monte Carlo experiment for an infinitely long time using a local algorithm, we would only observe a small fraction of all N-step self-avoiding walks.

Theorem 9.4.2 *Let $d = 2$ or 3, and let k be a positive integer. Let $CLEC_{k,N}$ be the cardinality of the largest ergodicity class of MAX(k) (for N-step walks). Then*

$$\limsup_{N\to\infty}(CLEC_{k,N})^{1/N} < \mu.$$

The proof of this theorem relies on Kesten's Pattern Theorem. The idea is that there are certain patterns that cannot be changed by k-site moves, and these patterns can occur many times on a self-avoiding walk. (Of course, the pattern depends on k.) A walk on which many such patterns occur must be in a small ergodicity class, since only some parts of the walk are able to change. But such patterns must occur many times on all but exponentially few walks, so those walks which are most able to change are necessarily in a small ergodicity class. The full proof is given in Section 9.7.2 for two dimensions. The proof will work in any dimension, as long as the existence of these special patterns is proven. This has been done in three dimensions by Madras and Sokal (1987), but it has not been done in four or more dimensions.

The practical implications of the nonergodicity (i.e. lack of irreducibility) of local algorithms are somewhat controversial. On the one hand, if your sole wish is to study "static" properties of a single self-avoiding walk (or a linear polymer), then the nonergodicity of local algorithms together with their long autocorrelation times (see below) should convince you to look at other algorithms. On the other hand, if you are interested in the *dynamic* properties of real polymers, then local moves are a better model for how real polymers move than are, say, the pivots of Section 9.4.3. Also, in more complicated systems (e.g. many polymers, or strong attractive interactions between monomers) other methods may be infeasible, and so one has little choice but to use local moves and hope that the systematic bias due to nonergodicity is negligible.

To conclude our discussion of local algorithms, we shall briefly discuss their autocorrelation times. Technically, they should be infinite, since nonergodicity prevents us from ever reaching the desired equilibrium distribution; so instead our discussion will apply either to the Markov chain whose state space is the ergodicity class of the straight walk, or to a Markov chain which allows self-intersecting walks (perhaps with reduced probability).

For each N-step walk ω, let $g(\omega)$ denote the mean distance between pairs of sites on ω:

$$g(\omega) = \frac{1}{N(N+1)} \sum_{i\neq j} |\omega(i) - \omega(j)|.$$

Then under the usual scaling assumption that the distribution of $g(\omega)$ scales like N^ν,

$$\tau_{int,g} \geq \text{const.} N^{2+2\nu}. \tag{9.4.1}$$

This follows from Corollary 9.2.3, since the variance $C_g(0)$ of $g(\omega)$ scales like $N^{2\nu}$, and since $|g(\omega) - g(\omega')| = O(1/N)$ whenever ω can be transformed into ω' by a local move. The same lower bound also holds for τ'_{exp}, by (9.2.27). It is generally believed that τ'_{exp} and τ_{exp} are in fact proportional to $N^{2+2\nu}$ for local algorithms that allow a wide enough class of moves [see Kremer and Binder (1988) for a discussion; note that their definition of τ differs from ours by a factor of N]. In the "mean-field" case of the VS algorithm applied to ordinary random walks (with $\nu = 1/2$), one can show that τ'_{exp} scales like $N^3 = N^{2+2\nu}$ [see Appendix 4.I of Doi and Edwards (1986)].

9.4.2 The "slithering snake" algorithm

A different kind of length-conserving dynamic algorithm was devised by Kron (1965) and by Wall and Mandel (1975) [see also Kron *et al.* (1967) and Mandel (1979)]. The basic move of the algorithm is to remove a bond from one end of the current walk while simultaneously trying to add a bond to the other end (rejecting the result if it is not self-avoiding). For an explicit description, use the following procedure as Step 2 in the Generic Fixed-Length Dynamic Algorithm.

> 2. Generate a random variable X which equals 0 with probability 1/2 and equals N with probability 1/2. If $X = 0$, then let Y be one of the $2d$ nearest neighbours of $\omega^{[t]}(0)$ (chosen uniformly at random), and set $\tilde{\omega} = (Y, \omega^{[t]}(0), \ldots, \omega^{[t]}(N-1))$. If $X = N$, then let Y be one of the $2d$ nearest neighbours of $\omega^{[t]}(N)$, and set $\tilde{\omega} = (\omega^{[t]}(1), \ldots, \omega^{[t]}(N), Y)$.

The nature of these moves has earned this algorithm and its variants the names "slithering snake" and "reptation" (the latter term is also used in polymer dynamics to describe similar motions of real polymers). This algorithm is reversible, but it is not irreducible: for example the walk of Figure 9.2 in Section 9.1 is frozen with respect to the slithering-snake algorithm in $\mathbf{Z}^2$. In fact, for sufficiently large N, it turns out that a positive fraction of all N-step walks are frozen, because there is a positive probability that both ends of the walk are "trapped" and cannot be extended by a single step in any direction. To be more precise, let Φ_N denote the set of all walks in $\mathcal{S}_N$ which are frozen with respect to the slithering-snake algorithm (that is, ω is in Φ_N if and only if the ergodicity class containing ω has cardinality

one). Using the terminology of Definitions 7.1.2 and 7.4.1, let P be a proper front pattern with the property that the $2d$ nearest neighbours of the first site of P are all sites of P. Let R be the walk whose sites are the sites of P in reverse order (see Figure 9.6; note that R is a proper tail pattern). Then any self-avoiding walk that begins with the pattern P and ends with the pattern R must be frozen; i.e. $\mathcal{S}_N(P,R) \subset \Phi_N$. Therefore (7.4.7) implies that

$$\liminf_{N\to\infty} \frac{|\Phi_N|}{c_N} > 0. \tag{9.4.2}$$

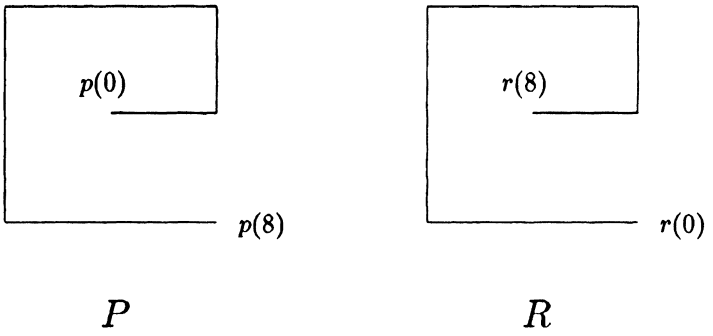

Figure 9.6: The proper front pattern $P = (p(0), \ldots, p(8))$ and the proper tail pattern $R = (r(0), \ldots, r(8))$. Any two-dimensional self-avoiding walk beginning with P and ending with R is frozen with respect to the slithering-snake algorithm.

Observe that although the intuitive description of the slithering-snake algorithm only involves moving one bond at a time, it is *not* a local algorithm by the definition of Section 9.4.1, because every site changes its position on the walk at every successful attempt [that is, $\omega^{[t]}(i)$ corresponds to $\omega^{[t+1]}(i \pm 1)$]. To emphasize the difference, we note that the analogue of (9.4.2) is false for local algorithms (since Kesten's Pattern Theorem 7.2.3 implies that most long walks contain many places where at least a single 1-site move can be made), and also that the analogue of Theorem 9.4.2 is false for the slithering-snake algorithm (since for example all N-step bridges are in the same ergodicity class as the straight self-avoiding walk, and $\lim_N (b_N/c_N)^{1/N} = 1$.) A better lower bound for the size of the largest ergodicity class is the following:

Proposition 9.4.3 *In the slithering-snake algorithm, denote by $\mathcal{E}_N$ the ergodicity class containing the N-step walk from the origin to $(N, 0, \ldots, 0)$. Then $|\mathcal{E}_N| \geq c_{2N}^{1/2}$.*

Remark. This bound is indeed better than $|\mathcal{E}_N| \geq b_N$ because $c_{2N}^{1/2} \geq \mu^N \geq b_N$ [by (1.2.10) and (1.2.17)].

Proof. Let $\mathcal{E}_N^*$ denote the set of all N-step walks in $\mathcal{S}_N$ which can be extended (possibly from both ends) to a $2N$-step self-avoiding walk; that is, ω is in $\mathcal{E}_N^*$ if and only if there is a walk $\varrho \in \mathcal{S}_{2N}$ such that ω occurs at some step of ϱ. Since every $2N$-step self-avoiding walk is the concatenation of two walks in $\mathcal{E}_N^*$, we see that $c_{2N} \leq |\mathcal{E}_N^*|^2$. Thus the proposition will be proved if we can show that $\mathcal{E}_N^*$ is contained in $\mathcal{E}_N$.

To complete the proof, let $\omega \in \mathcal{E}_N^*$, and let $\varrho \in \mathcal{S}_{2N}$ such that ω occurs at some step of ϱ. Let $\varrho(j)$ be the lexicographically largest site of ϱ [so ϱ lies in the half-space $x_1 \leq \varrho_1(j)$]. Now let υ be the self-avoiding walk $(\varrho(j), \ldots, \varrho(j+N))$ if $j \leq N$, or $(\varrho(j-N), \ldots, \varrho(j))$ if $j > N$ (so υ is an N-step subwalk of ϱ and has $\varrho(j)$ as an endpoint). Observe that ω and υ are in the same ergodicity class, since ω can be transformed into υ by "slithering" along the path ϱ. Since υ lies in the half-space $x_1 \leq \varrho_1(j)$ and has one endpoint at $\omega(j)$ on the boundary of this half-space, it can be transformed into the straight walk whose endpoints are $\omega(j)$ and $\omega(j) + (N, 0, \ldots, 0)$. Therefore υ is in $\mathcal{E}_N$, and hence so is ω. This completes the proof. □

Proposition 9.4.3 and (9.4.2) imply that for sufficiently large N, the cardinality of the largest ergodicity class of the slithering-snake algorithm on $\mathcal{S}_N$, $CLEC_{SS,N}$, satisfies

$$aN^{-(\gamma-1)/2} \leq \frac{CLEC_{SS,N}}{c_N} \leq 1 - \epsilon$$

for some positive constants a and ϵ. Of course, the lower bound is only rigorous if we can prove the expected scaling behaviour $c_N \sim A\mu^N N^{\gamma-1}$. We do know that γ exists and equals 1 in five or more dimensions (see Section 6.1), so there $CLEC_{SS,N}/c_N$ stays bounded away from both 0 and 1; it is not known whether this ratio goes to 0 in 2, 3, or 4 dimensions.

9.4.3 The pivot algorithm

The preceding dynamic algorithms only attempt to move a few bonds at a time. In contrast, the *pivot algorithm* attempts to move large pieces of the walk at every iteration. These big moves are more likely to be rejected

than are local moves, but a success is typically rewarded by a large change in global observables such as end-to-end distance.

The pivot algorithm picks a "pivot site" at random on the current walk, breaks the walk into two pieces at that site, and then applies a randomly chosen symmetry operation of $\mathbf{Z}^d$ to one piece, using the pivot site as the origin. As usual, the result is accepted if and only if it is self-avoiding. This algorithm was originally used by Lal (1969), and has subsequently been rediscovered by several authors (see the Notes at the end of this chapter). As we shall see, the pivot algorithm is remarkably efficient for the investigation of global observables: it requires about $O(N \log N)$ computer time to generate an "effectively independent" observation. (This is about as good as one has the right to expect, since it takes time $O(N)$ just to write down an N-step walk!)

To give a formal description of the pivot algorithm, let us first consider the symmetry group of $\mathbf{Z}^d$. To be precise, let $\mathcal{G}_d$ be the set of orthogonal linear transformations of $\mathbf{R}^d$ which leave the lattice $\mathbf{Z}^d$ invariant. In two dimensions, $\mathcal{G}_2$ has eight members: two axis reflections, two diagonal reflections, rotations by $\pm\pi/2$ and π, and the identity. For general d, a transformation g in $\mathcal{G}_d$ is completely determined by its action on the d positive unit vectors $e_1, \ldots, e_d$ of $\mathbf{Z}^d$. Since each $g(e_i)$ must be a unit vector of $\mathbf{Z}^d$, g can be uniquely specified by a permutation π of $\{1, \ldots, d\}$ and numbers $\epsilon_1, \ldots, \epsilon_d = \pm 1$ via the relations

$$g(e_i) = \epsilon_i e_{\pi(i)}. \tag{9.4.3}$$

Thus $\mathcal{G}_d$ has $2^d d!$ members. Next, observe that each g in $\mathcal{G}_d$ leaves the origin fixed (since g is a linear transformation). For every g in $\mathcal{G}_d$ and x in $\mathbf{Z}^d$, define g_x to be the corresponding affine transformation that leaves x fixed, i.e.

$$g_x(y) = g(y - x) + x \qquad \text{for every } y \in \mathbf{Z}^d.$$

We can now describe the basic version of the pivot algorithm by using the following Step 2 in the Generic Fixed-Length Dynamic Algorithm.

> 2. Choose an integer I uniformly at random from $\{0, 1, \ldots, N-1\}$. Set $x = \omega^{[t]}(I)$ (the "pivot site"). Choose a G uniformly at random from $\mathcal{G}_d$. Set $\tilde{\omega}(l) = \omega^{[t]}(l)$ for every $l \le I$ and $\tilde{\omega}(l) = G_x(\omega^{[t]}(l))$ for every $l > I$.

As we shall see, this procedure is reversible and irreducible. We can get variants of this algorithm if we choose I or G from some nonuniform distribution. We shall also discuss irreducibility and reversibility of these variants below. As a different kind of variant, we could always pivot the

shorter part of the walk, leaving the longer part fixed. This should improve the efficiency of the algorithm without changing the Markov chain in any important way.

It is not hard to check reversibility with respect to the uniform distribution on $\mathcal{S}_N$. Suppose that ω and ω' are distinct self-avoiding walks such that $P(\omega,\omega') > 0$. There could be several ways to get from ω to ω': specifically, suppose that there are m possible pairs $(i^{(j)}, g^{(j)})$ $(1 \le j \le m)$ such that applying the operation $g^{(j)}$ to ω with pivot site $\omega(i^{(j)})$ will produce ω'. Then

$$P(\omega,\omega') = \sum_{j=1}^{m} \Pr\{I = i^{(j)}\} \Pr\{G = g^{(j)}\}.$$

Observe that applying the operation $(g^{(j)})^{-1}$ to ω' with pivot site $\omega'(i^{(j)})$ will produce ω. Therefore, we see from the above equation that $P(\omega,\omega') = P(\omega',\omega)$ in the original algorithm, as well as in any variant that satisfies

$$\Pr\{G = g\} = \Pr\{G = g^{-1}\} \qquad \text{for every } g \text{ in } \mathcal{G}_d.$$

We shall now consider the irreducibility of the pivot algorithm and also of variants which choose I and G from possibly nonuniform distributions. First of all, since the angle between the i-th and $(i+1)$-th step of the walk can only change when $I = i$, such a variant cannot be irreducible unless we require $\Pr\{I = i\} > 0$ for every $i = 1, \ldots, N-1$. Also, if $\Pr\{I = 0\} = 0$, then irreducibility fails because the direction of the first step never changes. (Of course, if the observables being measured are invariant with respect to the symmetries of the lattice, then it cannot hurt to take $\Pr\{I = 0\} = 0$.) Thus the interesting questions about irreducibility of the variants arise when some symmetries are allowed to have zero probability. The following result holds in every dimension $d \ge 2$.

Theorem 9.4.4 *The pivot algorithm is irreducible, as is any variant which gives nonzero probability to all d reflections through coordinate hyperplanes $x_i = 0$ and to all rotations by $\pm\pi/2$ (which leave $d-2$ axes fixed). In fact, any walk in $\mathcal{S}_N$ can be transformed into a straight walk by some sequence of at most $2N-1$ such pivots.*

The proof will be given in Section 9.7.3. The basic idea is that if we consider a snug box around a walk, then we can try to "unfold" the walk by performing a reflection through one of the faces of the box.

The above theorem remains true if we replace $\pm\pi/2$ rotations by any set of symmetries that contains, for every distinct i and j in $\{1, \ldots, d\}$, a symmetry that sends e_i to e_j and another that sends e_i to $-e_j$ (for example, the set of all reflections through hyperplanes $x_i = x_j$ or $x_i = -x_j$). The

proof is the same. It is clear that some such set of symmetries must be used; notice that if we *only* allowed reflections through coordinate hyperplanes, then we could never change the angle between consecutive steps, and so the total number of right-angle turns in the walk could never change (in particular, straight walks would be frozen).

Some additional results about irreducibility of variants in two dimensions are known. If a variant gives nonzero probability to the three rotations $\pm\pi/2$ and π, then it is irreducible [see Section 3.5 of Madras and Sokal (1988)]. A variant is *not* irreducible if we only allow rotations by π (since the number of right-angle turns cannot change) or if we only allow rotations by $\pm\pi/2$ [a counterexample for $N = 223$ is shown on p. 139 of Madras and Sokal (1988)]. Finally, if we only allow the two diagonal reflections, then we *do* have irreducibility—in fact, any walk in $\mathcal{S}_N$ having exactly k right-angle turns can be transformed into a straight walk by some sequence of k diagonal reflections [Madras, Orlitsky, and Shepp (1990)]. As a consequence of this last result, we have

Corollary 9.4.5 *Let $d = 2$. For the transition probability P of the original pivot algorithm, $P^{2N-1}(\omega,\omega') > 0$ for every ω and ω' in $\mathcal{S}_N$.*

This means that the "diameter" of the state space of the two-dimensional pivot algorithm is at most $2N-1$ ($N-1$ pivots to straighten out ω, 1 pivot at the origin, and then $N-1$ to make ω').

Now that we have seen that the pivot algorithm is a *valid* method (since it is reversible and irreducible), it is is time to discuss why it is a *good* algorithm. Only a limited part of this discussion will be based on rigorous proofs; the rest will consist of nonrigorous arguments (scaling theory, etc.) supported by numerical evidence from computer experiments.

The intuitive picture, which we shall elaborate upon below, is the following. Firstly, since a pivot makes a large-scale change in a walk, it is reasonable to expect that we will obtain an "effectively independent" configuration (at least with respect to global observables) after relatively few successful pivots. It will turn out that "relatively few" means about $\log N$. Secondly, the probability of a particular pivot being accepted will tend to 0 as $N \to \infty$, but as some power law N^{-p}. Since there are no frozen configurations, this probability cannot decay faster than N^{-1}, and so $0 \le p \le 1$. (Numerically, p is estimated to be about 0.19 in two dimensions and 0.11 in three.) Thus one expects a successful pivot in every N^p attempts. Recalling the discussion following (9.2.11), we infer from these first two points that the integrated autocorrelation time for a global observable should be about $N^p \log N$. Finally, we also have to include the average amount of computer time required per attempted pivot. The amount of work—checking for intersections, updating arrays, etc.—is at worst proportional to N; so

suppose that the amount of computer time per attempt is on the order of N^q. Therefore the amount of computer time required per *successful* attempt is N^{p+q}, and the amount of computer time required per "effectively independent" observation of a global observable is $N^{p+q} \log N$. We shall argue below that $p + q = 1$.

In the remainder of the section we shall elaborate on the intuitive argument described above. As a guide for the first part, which says that relatively few successful pivots are needed to get an "effectively independent" observation of a global observable, we can consider a simpler model: the pivot algorithm applied to ordinary random walk. That is, the state space is now $\mathcal{S}_N^o$ (the set of all $(2d)^N$ ordinary walks), and the pivot algorithm now does not care about self-avoidance (so in Step 3 of the Generic Algorithm, we always set $\omega^{[t+1]} = \tilde{\omega}$). For this model, we can do exact calculations to prove rigorously that the integrated autocorrelation time $\tau^o_{int,g}$ for the global observable $g(\omega) = |\omega(N)|^2$ is asymptotic to $2 \log N$ as $N \to \infty$ (see Proposition 9.7.1 in Section 9.7.3). The same conclusion holds (except for a constant factor) for the global observables $\omega_i(N)$ and the squared radius of gyration [Madras and Sokal (1988)].

It is important to observe that the situation is quite different for the *exponential* autocorrelation times of the ordinary random walk: in particular, $\tau^o_{exp,g}$ is asymptotically equal to N as $N \to \infty$ for the global observable $g(\omega) = |\omega(N)|^2$ (Proposition 9.7.1). In fact, the exponential autocorrelation time for the entire chain, τ^o_{exp}, is also asymptotically proportional to N [Madras and Sokal (1988)]. It is easy to understand the situation for *local* observables: consider for example the angle between the 15-th and 16-th steps. The probability that this changes in a particular pivot is $1/N$ times a constant, since the angle can only change when the pivot site is $\omega(15)$, which happens with probability $1/N$. So both the integrated and exponential autocorrelation times for this observable should behave like N.

To summarize: global characteristics of walks tend to correspond to short modes of this system, while the long modes tend to be orthogonal to the quantities of interest. This emphasizes how the pivot algorithm is specially designed for looking at global quantities. It is reasonable to expect this to carry over to the self-avoiding case as well, and results of simulations seem to indicate that this is indeed what happens. However, proving such claims rigorously remains an open and apparently difficult problem.

We now turn to the amount of computer time required per attempted pivot, and its behaviour as N increases. The main issue is how long it takes to discover whether or not the proposed walk $\tilde{\omega}$ is self-avoiding. If we compute all of $\tilde{\omega}$ and then check for intersections, then each attempted pivot requires time proportional to N. But we can do better by looking for self-intersections as we compute $\tilde{\omega}$, so that we can stop early if one is

found. We expect that $\tilde{\omega}$ is most likely to intersect itself in the vicinity of the pivot site, so we first compute $\tilde{\omega}$ at the pivot site, and then move outwards towards both ends of the walk simultaneously, computing $\tilde{\omega}$ and checking for self-intersections as we go. We shall now make the description of this procedure more precise. In doing so, it will be convenient to use the following notation for integers $a \le b$ satisfying $a \le N$ and $b \ge 0$:

$$\omega[a,b] \equiv (\omega(\max\{a,0\}), \omega(\max\{a,0\}+1), \ldots, \omega(\min\{b,N\})).$$

Consider the following procedure for a single attempt of the pivot algorithm (where $\omega^{[t]}$ is the current walk).

(a) Choose the pivot site I and the symmetry G at random. Set $x = \omega^{[t]}(I)$, $j = 1$, and $\tilde{\omega}(I) = \omega^{[t]}(I)$.
(b) Set $\tilde{\omega}(I+j) = G_x(\omega^{[t]}(I+j))$ (if $I+j \le N$) and set $\tilde{\omega}(I-j) = \omega^{[t]}(I-j)$ (if $I-j \ge 0$).
(c) If $I+j \le N$, then check to see if $\tilde{\omega}(I+j)$ is in the set of sites $\tilde{\omega}[I-j+1, I+j-1]$. If it is, then the current attempt fails, so stop; otherwise, continue.
(d) If $I-j \ge 0$, then check to see if $\tilde{\omega}(I-j)$ is in the set of sites $\tilde{\omega}[I-j+1, I+j]$. If it is, then the current attempt fails, so stop; otherwise, continue.
(e) If $j < \max\{N-I, I\}$, then increase j by one and go to Step (b). Otherwise, the current attempt has succeeded, so set $\omega^{[t+1]} = \tilde{\omega}$ and stop.

Steps (a) and (b) can be performed in time $O(1)$ (i.e. independent of N). In addition, Steps (c) and (d) can also be performed in average time $O(1)$ with the use of a bit map or a hash table (see the discussion at the end of Section 9.1), as follows. We begin with an empty bit map (or hash table); at each step, it will contain the sites of $\tilde{\omega}$ that have already been computed. As each new site of $\tilde{\omega}$ is computed, we check to see whether its location is still vacant in the bit map; if so, then we add this site to the bit map, but otherwise we stop because we have found a self-intersection. In the case of a success, Step (e) requires time $O(N)$ for recording $\omega^{[t+1]}$ and reinitializing the bit map. In summary, we see that the total amount of work is proportional to the number of times that Step (b) is performed (i.e. the number of times through the "loop"). Define the random variable $H(\omega)$ to be the smallest value of j such that $\tilde{\omega}[I-j, I+j]$ is not self-avoiding (and set $H(\omega) = N$ if $\tilde{\omega}[0,N]$ is self-avoiding). Thus the amount of work per attempt is of order $E(H(\omega))$. Evidently this is at most $O(N)$, but we can improve this bound by the following heuristic argument. First we have

$$\Pr\{H(\omega) > k\} = \Pr\{\tilde{\omega}[I-k, I+k] \text{ is self-avoiding}\}$$

$$\approx \Pr\{\text{ a } 2k\text{-step self-avoiding walk pivoted at its midpoint is again self-avoiding}\}$$
$$\sim \text{const.} k^{-p}$$

where p is the exponent discussed above. We can now estimate the expectation of $H(\omega)$:

$$E(H(\omega)) = \sum_{k=0}^{N} \Pr\{H(\omega) > k\} \approx N^{1-p}.$$

Therefore the average amount of work per attempt is of order N^{1-p}, and so $p+q = 1$ as anticipated. This heuristic argument does in fact agree with computational experience.

This completes our discussion of why the integrated autocorrelation times for global observables are believed to be $O(N \log N)$ for the pivot algorithm.

9.5 Variable-length dynamic methods

In this section we shall discuss two dynamic methods whose state spaces include self-avoiding walks of various lengths. The Berretti-Sokal algorithm is the conceptually simplest such method: its state space is the set of all self-avoiding walks. The "join-and-cut" algorithm has as its state space the set of all pairs of self-avoiding walks whose lengths sum to some fixed number N. A third method, the BFACF algorithm, will be discussed in Section 9.6.1: its state space is the set of all self-avoiding walks with specified endpoints 0 and x for some fixed point x in $\mathbf{Z}^d$.

When using variable-length methods, the statistical analysis of the data can be more complicated than our discussion in Section 9.2 indicated. In that section, we assumed that the estimates from different values of N were independent. While this is true for fixed-length methods, where different values of N correspond to different simulations, it will be false for the algorithms of the present section. Berretti and Sokal (1985) show how to use maximum-likelihood estimation for their variable-length algorithm; the techniques developed there can be adapted to other algorithms.

9.5.1 The Berretti-Sokal algorithm

The Berretti-Sokal algorithm is designed to sample from the set of all self-avoiding walks of all possible lengths. It will be defined precisely below, but the basic idea is that at each step you either delete the last bond of

the walk or else you attempt to increase the length of the walk by adding a bond to the end (rejecting the attempt if the result is not self-avoiding). The state space is

$$\mathcal{S} \equiv \bigcup_{N=0}^{\infty} \mathcal{S}_N,$$

which is infinite, so we cannot ask for uniform probabilities on all walks. It is natural, however, to ask for uniform probabilities within each $\mathcal{S}_N$. The Berretti-Sokal algorithm simulates walks in the "canonical ensemble" (in contrast to the fixed-length "microcanonical ensemble"). This requires a parameter $z > 0$ (as in the Redner-Reynolds algorithm of Section 9.3.3). Each N-step self-avoiding walk is given a weight (i.e. a relative probability) of z^N. The sum of all the weights of walks in $\mathcal{S}$ is just the susceptibility $\chi(z)$. Using this weight to normalize the probabilities, we obtain the probability distribution

$$\pi(\omega) \equiv \pi_z(\omega) = \frac{z^{|\omega|}}{\chi(z)}. \tag{9.5.1}$$

Of course, this only makes sense if $\chi(z)$ is finite, so we shall henceforth assume that

$$0 < z < z_c = \mu^{-1}.$$

(In physical terminology, z is the "fugacity per bond", and $\chi(z)$ plays the role of a "partition function"; also, π is a "Gibbs distribution".) Observe that π is a genuine probability distribution on $\mathcal{S}$. The mean square displacement of a walk chosen at random from this distribution is

$$\sum_{\omega \in \mathcal{S}} |\omega(|\omega|)|^2 \pi(\omega) = \sum_{\omega \in \mathcal{S}} \frac{|\omega(|\omega|)|^2 z^{|\omega|}}{\chi(z)} = \xi_2(z)^2,$$

which is the square of the correlation length of order 2. Thus we can obtain information about the critical exponent ν_2, which is believed to equal ν. Moreover, the canonical ensemble is a natural setting for studying μ and γ, since the fraction of time that the Markov chain spends in $\mathcal{S}_N$ (i.e. the fraction of time that an N-step self-avoiding walk is observed) is

$$\sum_{\omega : |\omega| = N} \pi(\omega) = \frac{c_N z^N}{\chi(z)} \sim \frac{A(\mu z)^N N^{\gamma - 1}}{\chi(z)}.$$

We shall use $\langle \cdot \rangle_z$ to denote expectation with respect to π_z. For future reference, we note that the mean length of a walk is

$$\langle N \rangle_z = \frac{\sum_{N=0}^{\infty} N c_N z^N}{\sum_{N=0}^{\infty} c_N z^N} \simeq (z_c - z)^{-1} \tag{9.5.2}$$

[under the usual scaling assumptions, arguing as we did for (1.3.11)]. In particular, the mean length diverges as z increases to z_c.

We now state the algorithm of Berretti and Sokal (1985).

Berretti-Sokal (B-S) Algorithm. This algorithm generates a Markov chain $\{\omega^{[t]}\}$ on the state space $\mathcal{S}$ which is reversible with respect to π_z.

1. Let $\omega^{[0]}$ be any self-avoiding walk in $\mathcal{S}$. Set $t = 0$.
2. Let $N = |\omega^{[t]}|$. Generate a random variable X which is $+1$ with probability $2dz/(1+2dz)$ and -1 with probability $1/(1+2dz)$. If $X = +1$, then go to Step 3; if $X = -1$, then go to Step 4.
3. Try to add a step to $\omega^{[t]}$: Choose one of the $2d$ nearest neighbours of $\omega^{[t]}(N)$ uniformly at random; call this point Y. If Y is not already a site of $\omega^{[t]}$, then set $\omega^{[t+1]} = (\omega^{[t]}(0), \ldots, \omega^{[t]}(N), Y)$; if Y is a site of $\omega^{[t]}$, then set $\omega^{[t+1]} = \omega^{[t]}$. Increase t by one and go to Step 2.
4. Delete the last step of $\omega^{[t]}$: If $N > 0$, then set $\omega^{[t+1]} = (\omega^{[t]}(0), \ldots, \omega^{[t]}(N-1))$; if $N = 0$, then set $\omega^{[t+1]} = \omega^{[t]}$ (the 0-step walk). Increase t by one and go to Step 2.

It is easy to see that the Markov chain corresponding to the B-S algorithm is irreducible: any N-step walk can be transformed into the 0-step walk in N iterations, and vice versa. Now let us check reversibility. Let ω be an N-step self-avoiding walk, and let ω' be an $(N+1)$-step self-avoiding walk which can be obtained by adding a single step to ω. Then

$$\pi(\omega)P(\omega,\omega') = \frac{z^N}{\chi(z)} \left(\frac{2dz}{1+2dz} \cdot \frac{1}{2d} \right)$$

and

$$\pi(\omega')P(\omega',\omega) = \frac{z^{N+1}}{\chi(z)} \frac{1}{1+2dz},$$

which implies that

$$\pi(\omega)P(\omega,\omega') = \pi(\omega')P(\omega',\omega).$$

For all other choices of distinct ω and ω', both sides of the above equation are 0. And of course the equation is trivial when $\omega = \omega'$. This proves reversibility with respect to π_z.

We now turn our attention to the autocorrelation times of the Berretti-Sokal algorithm. Before summarizing what is rigorously known, we shall give a heuristic argument which provides a pretty good intuition for what

is happening. The first claim is that the autocorrelation times should be of the same order as the average time required to reach the 0-step walk from a typical initial walk in the state space. This is because before the 0-step walk is reached, the Markov chain still remembers the first steps of the initial walk, but the chain forgets everything once the 0-step walk is reached. Next, consider the process $N(t) \equiv |\omega^{[t]}|$, i.e. the length of the walk at time t. One expects this process to behave more or less like a random walk on the nonnegative integers having transition probabilities

$$\begin{aligned} P(i, i+1) &= \frac{2dz}{2dz+1} \frac{1}{2d} \mu = \frac{\mu z}{2dz+1} \\ P(i, i) &= \frac{2dz - \mu z}{2dz+1} \\ P(i, i-1) &= \frac{1}{2dz+1} \end{aligned}$$

for moderately large i (the factor μ in the first line is an approximation of c_{i+1}/c_i, the number of ways in which an average i-step self-avoiding walk can be extended by a single step). This random walk has a drift of $(\mu z - 1)/(2dz+1)$, which is *negative*. Thus the expected time for the process to go from a state N_0 to the state 0 is about N_0 divided by the magnitude of the drift. Finally, suppose that the initial walk $\omega^{[0]}$ is drawn at random from the equilibrium distribution π; then the expected time to reach 0 is about

$$\langle N(0) \rangle_z \frac{2dz+1}{1-\mu z};$$

by (9.5.2), this is asymptotically proportional to $\langle N \rangle_z^2$ as $z \to z_c = \mu^{-1}$. Thus we conclude from our heuristic argument that τ_{exp} should scale like $\langle N \rangle_z^2$ [i.e. like $(z_c - z)^{-2}$].

This argument does quite well in several respects. First, one can do exact calculations when the B-S algorithm is applied to ordinary random walks [for which the state space is $\cup_N \mathcal{S}_N^o$, and we take $0 < z < z_c = (2d)^{-1}$]. In this case, $N(t)$ is *exactly* a random walk with drift, and the integrated autocorrelation time of this observable can be shown to scale like $\langle N \rangle_z^2$ [see Appendix A of Berretti and Sokal (1985)]. Secondly, the random-walk-with-drift approximation is in fact a lower bound for the actual chain: an application of Corollary 9.2.3 (with $g = N$, $A = 1$, and the assumption that the probability distribution of N, in particular its standard deviation, scales like $\langle N \rangle_z$) shows that

$$\tau_{int,N} \geq \text{const.} \langle N \rangle_z^2; \tag{9.5.3}$$

by (9.2.27), this is also a lower bound for τ'_{exp}. Thirdly, Sokal and Thomas (1989) proved a rigorous upper bound, subject to the assumption that c_N

scales like $\mu^N N^{\gamma-1}$, that

$$\tau'_{exp} \le \text{const.} \langle N \rangle_z^{1+\gamma} . , \tag{9.5.4}$$

The exponent $1+\gamma$ is near 2 in all dimensions (and in fact equals 2 when $d \ge 5$; see Section 6.1), so the above two bounds place pretty narrow limits on the scaling behaviour of τ'_{exp}. The exact behaviour remains an open question. (We remark that the proof of Sokal and Thomas also works for the B-S algorithm applied to ordinary random walks, where $\gamma = 1$.)

Lastly, we mention a slightly weaker bound derived by Lawler and Sokal (1988), using very different methods:

$$\tau'_{exp} \le \text{const.} \langle N \rangle_z^{2\gamma} . \tag{9.5.5}$$

Their main tool is a general version of Cheeger's inequality, which in its original form was a lower bound on the second smallest eigenvalue of the Laplacian on a compact Riemannian manifold [Cheeger (1970)]. Cheeger's inequality has recently found a wide range of applications in problems involving rates of convergence to equilibrium in Markov chains [see Diaconis and Stroock (1991) and references therein, as well as in Lawler and Sokal (1988)].

Finally, we note that one could implement a variant of the B-S algorithm in which one is allowed to add or delete steps from *either* end of the walk. We can regard this as a combination of the B-S and the "slithering snake" algorithm. The resulting algorithm should behave very much like the B-S algorithm. A form of this variant was used by Kron *et al.* (1967).

9.5.2 The join-and-cut algorithm

The join-and-cut algorithm was invented by Caracciolo, Pelissetto, and Sokal (1992) as an efficient method for estimating the exponent γ. This algorithm works on a rather different state space: the set of all *pairs* of self-avoiding walks whose combined length is fixed. To formalize the definition, let M be a fixed positive integer. We define $\mathcal{T}_M$ to be the set of all pairs (ψ, φ) of self-avoiding walks such that $|\psi| + |\varphi| = M$:

$$\mathcal{T}_M \equiv \bigcup_{m=0}^{M} \mathcal{S}_m \times \mathcal{S}_{M-m} .$$

We shall see that the equilibrium distribution of the algorithm is uniform on $\mathcal{T}_M$, and hence the distribution of the length of the first walk in the pair is

$$\Pr\{|\psi| = k\} = \frac{c_k c_{M-k}}{|\mathcal{T}_M|} \approx k^{\gamma-1} (M-k)^{\gamma-1} ,$$

from which one can try to estimate γ.

The algorithm is as follows.

The Join-and-Cut Algorithm. This algorithm generates a Markov chain $\{X^{[t]}\} = \{(\psi^{[t]}, \varphi^{[t]})\}$ on the state space $\mathcal{T}_M$ which is reversible with respect to the uniform distribution on $\mathcal{T}_M$.

1. Let $X^{[0]} = (\psi^{[0]}, \varphi^{[0]})$ be any pair of self-avoiding walks in $\mathcal{T}_M$. Set $t = 0$.
2. Apply one iteration of the pivot algorithm (see Section 9.4.3) to $\psi^{[t]}$, obtaining $\hat{\psi}$. Then apply one iteration of the pivot algorithm to $\varphi^{[t]}$, obtaining $\hat{\varphi}$. (Alternatively, with the hope of reducing autocorrelation times, we could replace "one iteration" by "some fixed number n_{piv} of iterations", and "pivot algorithm" by "some length-conserving ergodic algorithm whose equilibrium distribution is uniform".)
3. (*Join*) Let $\zeta = \hat{\psi} \circ \hat{\varphi}$ be the concatenation of $\hat{\varphi}$ to $\hat{\psi}$.
4. (*Cut*) Choose J uniformly at random from $\{0, \ldots, M\}$. Set $\psi' = (\zeta(0), \ldots, \zeta(J))$ and $\varphi' = (\zeta(J), \ldots, \zeta(M))$. If both ψ' and φ' are self-avoiding, then set $\psi^{[t+1]} = \psi'$ and $\varphi^{[t+1]} = \varphi'$; otherwise, set $\psi^{[t+1]} = \hat{\psi}$ and $\varphi^{[t+1]} = \hat{\varphi}$.
5. Increase t by one and go to Step 2.

We emphasize that in Step 3 one does not need to check whether the walk ζ is self-avoiding. For purposes of comparison, however, let us consider also a variant of the join-and-cut algorithm in which we do perform this check. Specifically, this variant is obtained by replacing Steps 3 and 4 by the following:

3′. (*Join*) Let $\zeta = \hat{\psi} \circ \hat{\varphi}$ be the concatenation of $\hat{\varphi}$ to $\hat{\psi}$. If ζ is self-avoiding, then go to Step 4; otherwise, set $\psi^{[t+1]} = \hat{\psi}$ and $\varphi^{[t+1]} = \hat{\varphi}$ and go to Step 5.

4′. (*Cut*) Choose J uniformly at random from $\{0, \ldots, M\}$. Set $\psi^{[t+1]} = (\zeta(0), \ldots, \zeta(J))$ and $\varphi^{[t+1]} = (\zeta(J), \ldots, \zeta(M))$.

(Observe that whenever Step 4′ is performed, the resulting $\psi^{[t+1]}$ and $\varphi^{[t+1]}$ are necessarily self-avoiding.)

The transition probability matrix P of the join-and-cut algorithm can be expressed as the product of two transition matrices P_a and P_b, which correspond respectively to Step 2 and to Steps 3 and 4 of the algorithm. To describe P_a and P_b more precisely, let Q be the transition matrix of the ergodic length-conserving algorithm used in Step 2 for single walks, defined with respect to the state space $\mathcal{S}$ of all self-avoiding walks (thus the

ergodic classes of Q are precisely the sets $\mathcal{S}_N$, and $Q(\omega, \omega') = 0$ whenever $|\omega| \neq |\omega'|$). Then

$$P_a((\psi_1, \varphi_1), (\psi_2, \varphi_2)) = Q(\psi_1, \psi_2) Q(\varphi_1, \varphi_2)$$

(for (ψ_i, φ_i) in $\mathcal{T}_M$, $i = 1, 2$). If the length-conserving algorithm is reversible (i.e. if Q is symmetric), then so is P_a; more generally, if the restriction of Q to each $\mathcal{S}_N$ has the uniform distribution as a stationary distribution, then the same is true for the restriction of P_a to each $\mathcal{S}_{N_1} \times \mathcal{S}_{N_2}$. To describe P_b, suppose that (ψ_1, φ_1) and (ψ_2, φ_2) are distinct members of $\mathcal{T}_M$ whose concatenations $\psi_1 \circ \varphi_1$ and $\psi_2 \circ \varphi_2$ are the same; then

$$P_b((\psi_1, \varphi_1), (\psi_2, \varphi_2)) = \frac{1}{M+1}.$$

All other entries of P_a are 0, except for those on the main diagonal (which represent either a rejection in Step 4 or else the choice $J = |\psi^{[t]}|$). Clearly P_b is symmetric.

Unfortunately, the product of two symmetric matrices is not in general symmetric; therefore, even if P_a is symmetric (as it is when we use the pivot algorithm in Step 2), the product $P_a P_b$ cannot be expected to be symmetric. Thus the join-and-cut algorithm is *not* reversible in general. However its equilibrium distribution is nevertheless uniform on $\mathcal{T}_M$, because both P_a and P_b have the constant vector as a left eigenvector, and hence so does their product. The failure of reversibility is due to the fact that a single iteration of the algorithm consists of two stages whose order matters: doing the pivoting followed by the join-and-cut steps. We remark that the variant of the join-and-cut algorithm corresponding to the transition matrix $P = \frac{1}{2} P_a + \frac{1}{2} P_b$ *would* be reversible (in this variant, at each iteration one randomly decides either to do Step 2 or else to do Steps 3 and 4).

It is easy enough to prove that the join-and-cut algorithm is irreducible, as follows. For any length N, let ρ_N be the N-step walk with $\rho_N(i) = (i, 0, \ldots, 0)$ for every i. Given any ψ in $\mathcal{S}_m$ and any φ in $\mathcal{S}_{M-m}$, there exists a T such that $Q^T(\psi, \rho_m) > 0$ and $Q^T(\varphi, \rho_{M-m}) > 0$ (assuming that the restriction of Q to $\mathcal{S}_m$ is aperiodic, as it is in the case of the pivot algorithm). Since it is possible to pick $J = m$ on T consecutive iterations, we see that $P^T((\psi, \varphi), (\rho_m, \rho_{M-m}))$ and $P^T((\rho_m, \rho_{M-m}), (\psi, \varphi))$ are both nonzero. The concatenation of ρ_m and ρ_{M-m} is ρ_M, which may be cut successfully at any point, so the irreducibility of the algorithm follows.

It is possible to get some insight into the efficiency of the join-and-cut algorithm by a combination of rigorous analysis, scaling arguments and numerical work. We shall only give a brief description of some of these results. The reader is referred to Caracciolo *et al.* (1992) for more details.

First, let us estimate W, the amount of computer work that is required for a typical attempt to join and cut (that is, for Steps 3 and 4). For a given $\hat{\psi}$, $\hat{\varphi}$, and J (as produced by Step 2 and subsequently by Step 4), let $n = |\hat{\psi}|$ and let $L = |J - n|$. Then the attempt to join and cut may be described as an attempt to transfer the last L steps of $\hat{\psi}$ to the front of $\hat{\varphi}$ if $J < n$ (or vice versa if $J > n$). Roughly speaking this is like an attempt to concatenate two independent self-avoiding walks of lengths L and $M - n$ (or L and n). Thus if we start looking for self-intersections at the joining point and work our way outwards, then the probability that we will not have found one before k steps of both walks have been checked is approximately the same as the probability that two independent k-step self-avoiding walks can be concatenated successfully:

$$\Pr\{W > k\} \approx \frac{c_{2k}}{c_k^2} \sim \text{const.} k^{-(\gamma-1)}. \tag{9.5.6}$$

We have not included any n-dependence in (9.5.6) because we expect it to disappear when we average over n (since n, $M - n$, and L all typically have order of magnitude M). Therefore the average amount of work required for Steps 3 and 4 should be

$$E(W) = \sum_{k=0}^{M} \Pr\{W > k\} \approx M^{2-\gamma}.$$

Recall from Section 9.4.3 that when applying the pivot algorithm to $\mathcal{S}_N$ the average work per pivot should scale like N^{1-p}. This implies that the expected amount of work for one complete iteration of the join-and-cut algorithm, in which Step 2 consists of doing some fixed number n_{piv} of pivots on each of $\psi^{[t]}$ and $\varphi^{[t]}$, is

$$n_{piv} M^{1-p} + M^{2-\gamma}.$$

By all evidence, $p < \gamma - 1$ in two and three dimensions, and so $1 - p > 2 - \gamma$; this implies that the most of the computer work in the join-and-cut algorithm is used in the pivoting step, even when $n_{piv} = 1$.

Suppose for the moment that n_{piv} is very large. Then the join-and-cut algorithm can be thought of as an "idealized algorithm", in which Step 2 actually produces walks $\hat{\psi}$ and $\hat{\varphi}$ that are independent of $\psi^{[t]}$ and $\varphi^{[t]}$. This idealized algorithm is more amenable to rigorous analysis: Caracciolo *et al.* (1992) prove that the exponential autocorrelation time is at most $M^{\gamma-1}$ (under the usual scaling assumption $c_N \sim A\mu^N N^{\gamma-1}$). This is done by showing that $\tau_{exp} \simeq M^{\gamma-1}$ for the variant that uses the idealized Step

2 in conjunction with Steps 3′ and 4′ described earlier, and then appealing to Proposition 9.2.4 (although some extra work is needed, since these algorithms are not reversible).

Now consider the original join-and-cut algorithm with $n_{piv} = 1$. Since the idealized algorithm should be more efficient than the actual algorithm with respect to the observable $n = |\psi^{[t]}|$, and more generally observables $g(n)$ that depend only on n, we shall define the exponent h by

$$\tau_{int,g(n)} \simeq M^{\gamma-1+h},$$

and we can expect that h is positive and hope that it is small. Combining this with the discussion above, we conclude that the amount of computer time per effectively independent observation scales like $M^{\gamma-1+h}M^{1-p}$, i.e. like M^{κ} where $\kappa \equiv \gamma - p + h$. Using the conjectured values of γ and p from (1.1.11) and Section 9.4.3 respectively, the bound $\kappa \geq \gamma - p$ becomes (approximately) $\kappa \geq 1.15$ in $\mathbf{Z}^2$, $\kappa \geq 1.05$ in $\mathbf{Z}^3$, and $\kappa \geq 1$ in four or more dimensions. Caracciolo *et al.* (1992) argue that in fact κ should equal 1 in four or more dimensions, which would virtually make this an optimal algorithm there. They also report the results of Monte Carlo runs which lead them to estimate that κ is about 1.5 in two dimensions, which is significantly better than the Berretti-Sokal algorithm [compare (9.5.3)].

9.6 Fixed-endpoint methods

This section will discuss some dynamic Monte Carlo methods that generate self-avoiding walks with endpoints that have been specified in advance.

First we shall describe the relevant state spaces. For each x in $\mathbf{Z}^d$ ($x \neq 0$), we denote by $\mathcal{S}_N(x)$ the set of all N-step self-avoiding walks ω having $\omega(0) = 0$ and $\omega(N) = x$. In this section, we shall always assume that N and $||x||_1$ have the same parity, since $\mathcal{S}_N(x)$ is empty otherwise. Also, we denote by $\mathcal{S}(x) = \cup_N \mathcal{S}_N(x)$ the set of all self-avoiding walks having endpoints 0 and x. When generating walks with fixed length and fixed endpoints, then we want to sample from the uniform distribution on $\mathcal{S}_N(x)$:

$$\pi(\omega) \equiv \pi_N^x(\omega) = \frac{1}{c_N(0,x)} \qquad \text{for every } \omega \text{ in } \mathcal{S}_N(x). \tag{9.6.1}$$

When generating walks from the variable-length fixed endpoint ensemble, the situation is similar to that of Section 9.5.1. In addition to specifying the endpoint x, we also specify a parameter z (the "fugacity per bond")

between 0 and $z_c = \mu^{-1}$. We sample from the Gibbs distribution

$$\pi(\omega) \equiv \pi_z^x(\omega) = \frac{1}{\Xi(z,x)}|\omega| z^{|\omega|} \qquad \text{for every } \omega \text{ in } \mathcal{S}(x), \tag{9.6.2}$$

where $\Xi(z,x)$ is the normalizing constant

$$\Xi(z,x) = \sum_{N=0}^{\infty} N z^N c_N(0,x) = z\frac{\partial}{\partial z} G_z(0,x). \tag{9.6.3}$$

The variable-length ensemble is the natural choice for studying the critical exponent α_{sing}, defined in (1.4.13) by

$$c_N(0,x) \sim B\mu^N N^{\alpha_{sing}-2}$$

This is because the fraction of time that the observed walk has length n is

$$\sum_{\omega \in \mathcal{S}_n(x)} \pi(\omega) = \frac{n c_n(0,x) z^n}{\Xi(z,x)} \sim \frac{B}{\Xi(z,x)} (\mu z)^n n^{\alpha_{sing}-1}, \tag{9.6.4}$$

and so we can estimate α_{sing} by fitting a distribution of this form to the observed data (for fixed z near μ^{-1} and fixed x).

We remark that the multiplicative factor $|\omega|$ in (9.6.2) is not there for any deep reason, but only because this is what the algorithm of Section 9.6.1 naturally gives. By modifying the algorithm, one could get a different π, but there does not appear to be a good reason to do so.

Recall from Definition 3.2.1 that when $||x||_1 = 1$, we can associate each walk in $\mathcal{S}_N(x)$ with an $(N+1)$-step self-avoiding polygon. Thus any of the methods discussed in this section can be used to study self-avoiding polygons simply by fixing x to be a nearest neighbour of the origin. In this case, we say that we are working with the ensemble of "rooted" polygons: there is a particular bond (the one joining x to the origin) which must occur in every polygon of the state space. It is also possible to work with the ensemble of "unrooted" polygons, where each bond of the current polygon is allowed to change during the iteration of the algorithm. Then the state space is the set of all polygons on the lattice (or their equivalence classes up to translation). There is little difference between the two ensembles in practice, aside from a factor of $N+1$ in their cardinalities [recall (3.2.1)] and the orientation of the rooted bond (which is irrelevant for most simulations). However, the Markov chains that are defined on the two ensembles are different in a non-trivial way; for example, a proof of irreducibility for the unrooted ensemble may not work for the rooted ensemble.

9.6.1 The BFACF algorithm

We shall first discuss an algorithm due to Berg, Foerster, Aragão de Carvalho, Caracciolo, and Fröhlich (for references, see the Notes at the end of the chapter). This algorithm uses transitions of a local nature to generate walks in the variable-length fixed endpoint ensemble according to the distribution given by (9.6.2) and (9.6.3).

The elementary transformations for this algorithm are depicted in Figure 9.7. Each transformation is determined by choosing a bond of the

Figure 9.7: The elementary transformations of the BFACF algorithm.

current walk [say the bond from $\omega(i)$ to $\omega(i+1)$] and one of the $2d-2$ lattice directions perpendicular to the bond (let e be a unit vector in the chosen direction). Let x and y denote the lattice points $\omega(i)+e$ and $\omega(i+1)+e$ respectively. The transformation then moves the chosen bond by unit distance in the direction e, so that its new endpoints are x and y. Then there are now three possibilities, as illustrated in Figure 9.8:

(a) if the original ω had $\omega(i-1) \neq x$ and $\omega(i+2) \neq y$, then we add two bonds: the new walk is $\tilde{\omega} = (\omega(0), \ldots, \omega(i), x, y, \omega(i+1), \ldots, \omega(|\omega|))$, and $|\tilde{\omega}| = |\omega| + 2$;

(b) if the original ω had $\omega(i-1) = x$ and $\omega(i+2) = y$, then we remove two bonds: the new walk is $\tilde{\omega} = (\omega(0), \ldots, \omega(i-1), \omega(i+2), \ldots, \omega(|\omega|))$, and $|\tilde{\omega}| = |\omega| - 2$;

(c) if the original ω had $\omega(i-1) \neq x$ and $\omega(i+2) = y$ [or, respectively, $\omega(i-1) = x$ and $\omega(i+2) \neq y$], then the new walk is $\tilde{\omega} = (\omega(0), \ldots, \omega(i), x, \omega(i+2), \ldots, \omega(|\omega|))$ [respectively, $\tilde{\omega} = (\omega(0), \ldots, \omega(i-1), y, \omega(i+1), \ldots, \omega(|\omega|))$]. Here, $|\tilde{\omega}| = |\omega|$.

[If the chosen bond is the first bond of the walk, then $i = 0$ and we always have $\omega(i-1) \neq x$. Similarly, $\omega(|\omega|+1) \neq y$ always.] We shall write ΔN to denote $|\tilde{\omega}| - |\omega|$, the change in the number of bonds of the walk for each possibility; we shall say that (a), (b), and (c) are $\Delta N = +2$, -2, and 0 transformations respectively.

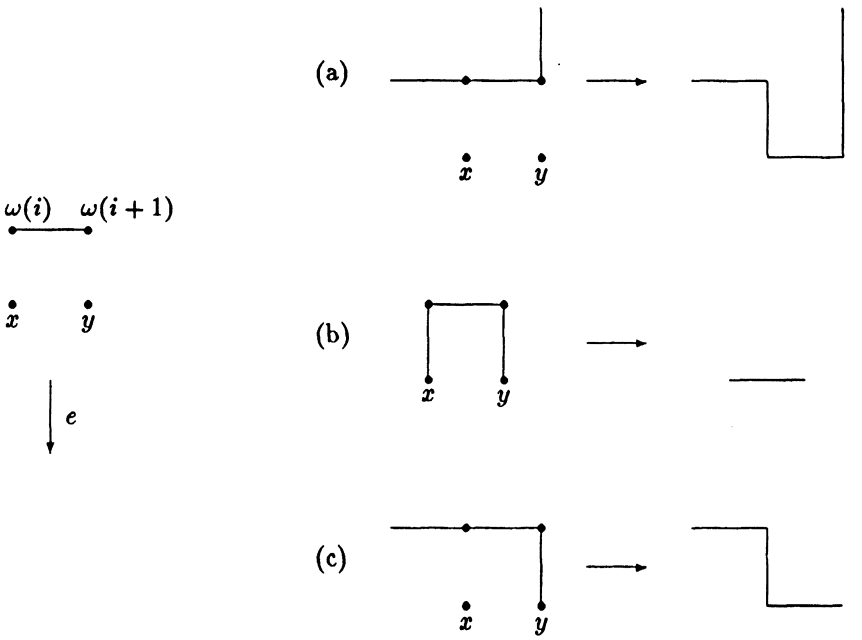

Figure 9.8: The three possibilities for a BFACF move, in detail.

To complete the definition of the BFACF algorithm, we need three numbers $p(+2)$, $p(-2)$, and $p(0)$ between 0 and 1 (they also must satisfy certain other conditions; see below).

BFACF Algorithm. This algorithm generates a Markov chain $\{\omega^{[t]}\}$ on the state space $\mathcal{S}(x)$.

1. Let $\omega^{[0]}$ be any walk in $\mathcal{S}(x)$. Set $t = 0$.
2. Choose an integer I uniformly at random from $\{0, 1, \ldots, |\omega^{[t]}| - 1\}$.
3. Consider the $2d-2$ walks $\tilde{\omega}$ that would be obtained by moving the I-th bond of $\omega^{[t]}$ in one of the directions perpendicular to the vector $\omega^{[t]}(I+1) - \omega^{[t]}(I)$. Choose one of these walks at random, with probabilities $p(|\tilde{\omega}| - |\omega^{[t]}|)$. (If these $2d-2$ probabilities add up to $q < 1$, then also choose $\tilde{\omega} = \omega^{[t]}$ with probability $1 - q$.)
4. If $\tilde{\omega}$ is self-avoiding, then set $\omega^{[t+1]} = \tilde{\omega}$; otherwise, set $\omega^{[t+1]} = \omega^{[t]}$.
5. Increase t by one and go to Step 2.

The necessary constraints on $p(+2)$, $p(-2)$, and $p(0)$ are given by the following lemma.

Lemma 9.6.1 *The BFACF algorithm is well-defined and is reversible with respect to π_z^x if and only if the following constraints are satisfied:*

$$p(+2) = z^2 p(-2), \tag{9.6.5}$$

$$p(+2) \le \frac{z^2}{1 + (2d-3)z^2}, \tag{9.6.6}$$

and

$$2p(0) + (2d-4)p(+2) \le 1. \tag{9.6.7}$$

Proof. First we consider reversibility. Suppose that ω and $\tilde{\omega}$ are distinct walks in $\mathcal{S}(x)$ such that $P(\omega, \tilde{\omega}) > 0$. On the one hand, if $|\omega|$ and $|\tilde{\omega}|$ differ by 2, then

$$P(\omega, \tilde{\omega}) = \frac{1}{|\omega|} p(|\tilde{\omega}| - |\omega|);$$

the condition (9.1.5) for reversibility in this case reduces to (9.6.5). On the other hand, if $|\omega| = |\tilde{\omega}|$, then

$$P(\omega, \tilde{\omega}) = \frac{2}{|\omega|} p(0),$$

since there are two possible choices of bond of ω that can produce $\tilde{\omega}$ [for example, in (c) of Figure 9.8, we get the same result by choosing the bond joining $\omega(i+1)$ to $\omega(i+2) = y$ and moving it in the direction $x - y$]; the reversibility condition imposes no additional constraint in this case.

Next, we note that the algorithm is well-defined if and only if the sum of the $2d-2$ probabilities in Step 3 does not exceed 1. There are several possibilities to consider, depending upon the relative orientations of the I-th, $(I-1)$-th and $(I+1)$-th bonds of $\omega^{[t]}$ (see Figure 9.9):

(i) All $2d-2$ directions yield $\Delta N = +2$: This requires $(2d-2)p(+2) \le 1$.
(ii) One direction yields $\Delta N = 0$, while the others yield $\Delta N = +2$: This requires $p(0) + (2d-3)p(+2) \le 1$.
(iii) Two directions yield $\Delta N = 0$, while the others yield $\Delta N = +2$: This requires $2p(0) + (2d-4)p(+2) \le 1$.
(iv) One direction yields $\Delta N = -2$, while the others yield $\Delta N = +2$: This requires $p(-2) + (2d-3)p(+2) \le 1$.

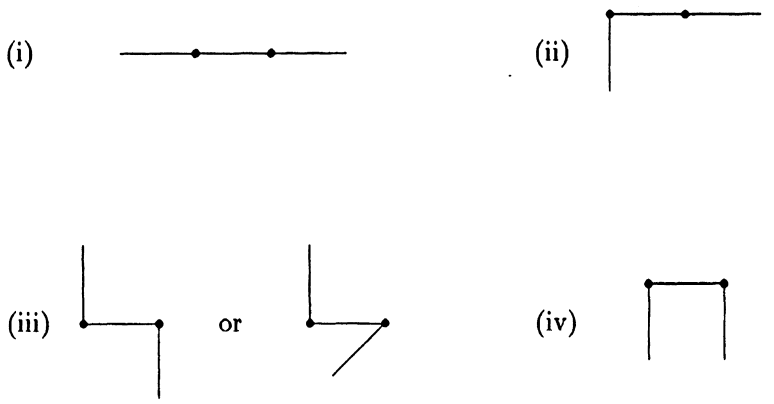

Figure 9.9: Proof of Lemma 9.6.1: relative orientations of three consecutive bonds.

The inequality of (ii) is redundant, since it follows from those of (i) and (iii). Next, substituting (9.6.5) into the inequality of (iv) gives

$$(z^{-2} + (2d-3))p(+2) \leq 1, \tag{9.6.8}$$

which is stronger than the inequality of (i) since $z \leq z_c < 1$. The inequality (9.6.8) is the same as (9.6.6), and the inequality of (iii) is the same as (9.6.7), so the lemma is proven. □

Now that we have a continuum of possible parameter values for a valid BFACF algorithm, we want to find the "best" choices of $p(+2)$, $p(-2)$, and $p(0)$ (for a given fixed z). Intuitively, we should prefer large values of these probabilities, so as to reduce the probability of "null transitions" (i.e. the quantity $1 - q$ described in Step 3 of the algorithm). Indeed, as we saw in Proposition 9.2.4 and the remark that follows it, increasing the off-diagonal elements of the transition matrix can only decrease the autocorrelation times (or at worst leave them unchanged). In two dimensions, the situation is easy: the constraint (9.6.7) simplifies to $p(0) \leq 1/2$, so the three probabilities can be maximized simultaneously:

$$p(+2) = z^2/(1+z^2), \quad p(-2) = 1/(1+z^2), \quad p(0) = 1/2. \tag{9.6.9}$$

In three or more dimensions, the constraint (9.6.7) forces a tradeoff between $p(0)$ and $p(+2)$. The standard choice is the point determined by the

intersection of the equalities corresponding to (9.6.6) and (9.6.7), which is

$$p(+2) = \frac{z^2}{1 + (2d-3)z^2}, \qquad p(-2) = \frac{1}{1 + (2d-3)z^2},$$

$$p(0) = \frac{1 + z^2}{2[1 + (2d-3)z^2]}. \tag{9.6.10}$$

Observe that setting $d = 2$ in (9.6.10) gives the values of (9.6.9). Caracciolo, Pelissetto, and Sokal (1990) have proven rigorously that (9.6.10) is close to optimal in every dimension.

Let us now turn to the problem of irreducibility. In two dimensions, the algorithm is irreducible for every $x \neq 0$ (see Theorem 9.7.2). In three dimensions, the algorithm is *not* irreducible if $||x||_\infty = 1$ (in particular, for the case of self-avoiding polygons). This is essentially because of knots: Consider the closed curve defined by the steps of the current walk of $\mathcal{S}(x)$ and by the line segment joining x to the origin. Each possible BFACF transformation may be viewed as the result of a continuous deformation of this closed curve during which it never crosses itself. In the terminology of topology, we say that the result of a BFACF transformation is *ambient isotopic* to the initial curve. Thus, for $||x||_\infty = 1$, the walks in any given ergodicity class of the BFACF algorithm must all correspond to the same knot type. The converse assertion, that the ergodicity classes correspond precisely to knot classes, has been proven by Janse van Rensburg and Whittington (1991) for the special case of unrooted polygons by showing that "Reidemeister moves" on knots can be achieved using BFACF moves. When $||x||_\infty \geq 2$ in three dimensions, then the BFACF algorithm is irreducible [Janse van Rensburg (1992a)].

We conclude our discussion of the BFACF algorithm with a look at its autocorrelation times. These tend to be large, and it is not hard to identify one of the reasons: the "area" determined by a walk is a very slow mode. (The meaning of "area" is obvious in the case $d = 2$, $||x||_1 = 1$; in general, consider a fixed walk ζ from 0 to x and let $a(\omega)$ be the minimum area of a lattice surface whose boundary is the union of ω and ζ.) The problem is that an N-step walk ω can have $a(\omega)$ of order N^2; since a single BFACF move can only change a by one unit, such a configuration can survive a very long time before being changed into something substantially different. In particular, we can apply Corollary 9.2.3 with $A = 1$ to obtain

$$\tau_{int,a} \geq \text{const.} \langle N \rangle^{4\nu} \tag{9.6.11}$$

(under the usual assumption that the probability distribution of a, and in particular its standard deviation, scales like $N^{2\nu}$). The slowness of a

to change for certain configurations was exploited further by Sokal and Thomas (1988), who proved the unsettling result that the exponential autocorrelation time of the BFACF algorithm is *infinite* (see Theorem 9.7.4).

9.6.2 Nonlocal methods

In Section 9.6.1, we saw that the BFACF algorithm has rather long autocorrelation times. Recalling that the pivot algorithm is more efficient than local algorithms in the free-endpoint ensemble (recall Section 9.4), it is clearly desirable to try to find large-scale transformations of self-avoiding walks that work in the fixed-endpoint ensemble. The transformations of the pivot algorithm of Section 9.4.3 do not leave both endpoints fixed, in general; however, other fixed-length transformations that use one or two "pivot sites" have been used with some success.

Fixed-length transformations have been used in the ensemble $\mathcal{S}_N(x)$ to study properties such as the radius of gyration or knottedness, particularly in the case of self-avoiding polygons ($||x||_1 = 1$). They have also been used in the variable-length ensemble $\mathcal{S}(x)$ together with BFACF moves in the hope of obtaining a more efficient algorithm for this ensemble.

We now describe fixed-length transformations which leave both endpoints fixed (see Figure 9.10). In these descriptions, ω is always an N-step self-avoiding walk.

1. *Inversion:* For integers k and l ($0 \le k < l \le N$), define the new walk $\tilde{\omega}$ by

$$\tilde{\omega}(i) = \begin{cases} \omega(k) + \omega(l) - \omega(k+l-i) & \text{if } k \le i \le l \\ \omega(i) & \text{otherwise.} \end{cases}$$

Thus the subwalk $(\tilde{\omega}(k), \ldots, \tilde{\omega}(l))$ is the inversion through the point $(\omega(k) + \omega(l))/2$ of the points $(\omega(l), \ldots, \omega(k))$. Another way to view inversion is by the sequence of bonds $\Delta\omega(i) \equiv \omega(i) - \omega(i-1)$. Then the bonds of $\tilde{\omega}$ are $\Delta\omega(1), \Delta\omega(2), \ldots, \Delta\omega(k), \Delta\omega(l), \Delta\omega(l-1), \ldots, \Delta\omega(k+2), \Delta\omega(k+1), \Delta\omega(l+1), \ldots, \Delta\omega(N)$.

2. *Cyclic permutation:* For an integer i ($0 < i < N$), define the new walk $\tilde{\omega}$ by breaking ω into two pieces at $\omega(i)$ and then concatening the two pieces in the other order. Thus the bonds of $\tilde{\omega}$ are $\Delta\omega(i+1), \Delta\omega(i+2), \ldots, \Delta\omega(N), \Delta\omega(1), \Delta\omega(2), \ldots, \Delta\omega(i)$.

3. *Lattice symmetries:* Using the notation of Section 9.4.3, let $g \in \mathcal{G}_d$ be a lattice symmetry. Let k and l be integers ($0 \le k < l \le N$) and let $x = \omega(k)$. If $g_x(\omega(l)) = \omega(l)$, then we get a new walk $\tilde{\omega}$ by applying this symmetry to

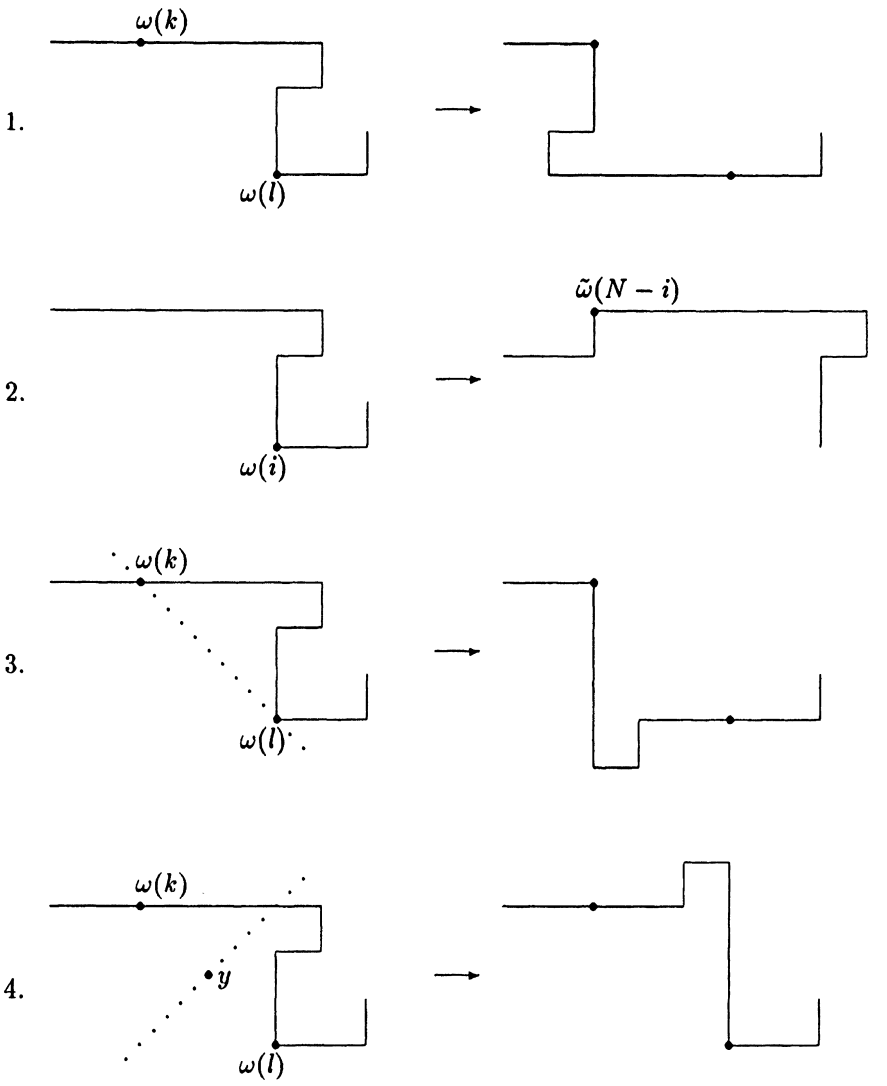

Figure 9.10: Length-preserving fixed-endpoint transformations: 1. inversion; 2. cyclic permutation; 3. reflection through line of slope -1; 4. reversing reflection through line of slope $+1$, where y is the midpoint between $\omega(k)$ and $\omega(l)$.

the part of ω between k and l:

$$\tilde{\omega}(i) = \begin{cases} g_x(\omega(i)) & \text{if } k \le i \le l \\ \omega(i) & \text{otherwise.} \end{cases}$$

Observe that the two "pivot sites" $\omega(k)$ and $\omega(l)$ are both fixed by g_x.

4. *Reversing lattice symmetries:* Again, let g be a lattice symmetry and let k and l be integers ($0 \le k < l \le N$). Now suppose that there exists a y such that $g_y(\omega(k)) = \omega(l)$ and $g_y(\omega(l)) = \omega(k)$. Then we can get a new walk $\tilde{\omega}$ by applying this symmetry to the part of ω between k and l, and reversing the order in which the sites appear in this part:

$$\tilde{\omega}(i) = \begin{cases} g_y(\omega(k+l-i)) & \text{if } k \le i \le l \\ \omega(i) & \text{otherwise.} \end{cases}$$

Dubins, Orlitsky, Reeds, and Shepp (1988) proposed (and proved the irreducibility of) an algorithm for unrooted polygons of fixed length in two dimensions. The DORS algorithm, as we shall call it, uses only inversion (1 above) and reversing diagonal reflection (4 above). The latter move may be described in words as follows. Choose two sites on the polygon such that the line segment L joining them makes an angle of $\pm\pi/4$ with the coordinate directions. Break the polygon into two pieces by cutting it at the two chosen sites, and reflect one of the pieces through the line which is the perpendicular bisector of L.

To prove that the DORS algorithm is irreducible, one shows first that inversions suffice to transform any polygon into a rectangle, and then that any rectangle may be transformed into any other rectangle by an inversion and a reversing reflection. The details of the proof are given in Section 9.7.4 (Theorem 9.7.3). Notice that an inversion does not change the number of bonds parallel to the x_1-axis, and so inversions alone do not suffice for ergodicity.

For the general fixed-length fixed-endpoint ensemble $\mathcal{S}_N(x)$ in two dimensions, the transformations of the DORS algorithm also provide an irreducible algorithm, but the proof is more involved [Madras, Orlitsky, and Shepp (1990)]. In higher dimensions, these transformations are not enough because if the initial walk is contained in the hyperplane $x_1 = 0$, say, then all of the resulting walks will lie in the same hyperplane.

To ensure irreducibility in $\mathcal{S}_N(x)$ in three or more dimensions, it suffices to use inversions (1 above), diagonal reflections (3 above), and reversing diagonal reflections (4 above). Here, a "diagonal reflection" is a reflection through a hyperplane which makes angles of $\pm\pi/4$ with two coordinate directions and angles of 0 with the remaining $d-2$ directions. Irreducibility is proven by a lengthy argument that uses induction on the number of

dimensions. The proof in d dimensions, even for $||x||_1 = 1$, requires knowledge of irreducibility of all fixed-length fixed-endpoint ensembles in $d-1$ dimensions. For details, see Madras *et al.* (1990).

Caracciolo, Pelissetto, and Sokal (1990) introduced an algorithm for the variable-length fixed-endpoint ensemble $\mathcal{S}(x)$, which uses inversion and cyclic permutation in addition to the usual BFACF transformations. This algorithm is irreducible in every dimension [Madras *et al.* (1990)].

9.7 Proofs

This section contains the longer proofs and calculations that have been deferred from the preceding sections of this chapter. The subsections may be read in any order.

9.7.1 Autocorrelation times

In this section we shall provide several arguments which were postponed from our discussion of the spectral theory of autocorrelation times that was begun in Section 9.2.3.

Let T be a self-adjoint contraction operator on $l^2(\pi)$. Then the spectrum $\sigma(T)$ is a subset of the interval $[-1,1]$. The Spectral Theorem [see for example Reed and Simon (1972)] tells us that there is a *spectral measure* E such that

$$T = \int_{[-1,1]} \lambda \, dE(\lambda);$$

in fact, for every positive integer k we have

$$T^k = \int_{[-1,1]} \lambda^k \, dE(\lambda).$$

Recall that for every Borel subset A of $[-1,1]$, $E(A)$ is a projection operator; in particular, $E(\emptyset) = 0$ and $E([-1,1]) = I$. Also, for every g in $l^2(\pi)$ we define E_g by

$$E_g(A) \equiv (g, E(A)g) = ||E(A)g||_2^2 \quad \text{for Borel sets } A \subset [-1,1].$$

Then E_g is a positive measure and

$$(g, T^k g) = \int_{[-1,1]} \lambda^k \, dE_g(\lambda) \tag{9.7.1}$$

for every positive integer k.

We can use this representation to prove

$$r(T) = \sup_f \limsup_{n\to\infty} |(f, T^n f)|^{1/n},$$

which is Equation (9.2.21). Let $q(T)$ denote the right hand side of the above equation. Since $r(T) = ||T||$, we clearly have $r(T) \geq q(T)$. Thus it suffices to prove the reverse inequality. Choose t so that $0 < t < r(T)$ and let $A[t] = \{\lambda : t < |\lambda| \leq 1\}$. We claim that there is a g in $l^2(\pi)$ such that $E_g(A[t]) > 0$. If not, then for every g

$$|(g, Tg)| \leq \left| \int_{-t}^{t} \lambda \, dE_g(\lambda) \right| \leq t E_g([-1,1]) = t||g||_2^2,$$

which contradicts $t < r(T) = ||T||$, so the claim is true. For g as in the claim, we have for any even n that

$$(g, T^n g) \geq \int_{A[t]} |\lambda|^n \, dE_g(\lambda) \geq t^n E_g(A[t]),$$

which implies that $q(T) \geq t$. Since this holds whenever $0 < t < r(T)$ we have $q(T) \geq r(T)$, so Equation (9.2.21) is proven.

We now return to our Markov chain, with $T = P - \Pi$ and E the corresponding spectral measure. Let g be a function in $l^2(\pi)$ and let $h = (I - \Pi)g$ be its projection onto the space of functions with mean 0. Since g and h differ by a constant, we know that $C_g(k) = C_h(k)$ for every $k \geq 0$. By (9.2.22) and (9.7.1),

$$C_g(k) = C_h(k) = \int_{[-1,1]} \lambda^k dE_h(\lambda) \quad \text{for every } k \geq 0$$

(where we interpret $0^0 = 1$). Using this in (9.2.10), along with the identity

$$\frac{1}{2} + \sum_{k=1}^{\infty} \lambda^k = \frac{1}{2}\left(\frac{1+\lambda}{1-\lambda}\right),$$

we obtain

$$\tau_{int,g} = \tau_{int,h} = \frac{\frac{1}{2}\int_{[-1,1]} \left(\frac{1+\lambda}{1-\lambda}\right) dE_h(\lambda)}{\int_{[-1,1]} dE_h(\lambda)}. \tag{9.7.2}$$

The support of E_h lies in $[-1, s]$, where $s = \sup \sigma(P - \Pi)$; together with (9.7.2) and the fact that $(1+\lambda)/(1-\lambda)$ is increasing for λ in $[-1, 1)$, this tells us that

$$\tau_{int,g} = \tau_{int,h} \leq \frac{1}{2}\left(\frac{1+s}{1-s}\right).$$

By (9.2.26), this proves (9.2.27).

We can now give a quick proof of Proposition 9.2.2 from Section 9.2.3, which says that $\tau_{int,g} \geq \frac{1}{2}(1+\rho_g(1))/(1-\rho_g(1))$, where $\rho_g(1) = C_g(1)/C_g(0)$.

Proof of Proposition 9.2.2. Let g be a nonconstant function in $l^2(\pi)$, and let $h = (I - \Pi)g$. Then E_h is a finite measure that is not identically 0, and so $E_h / \int dE_h(\lambda)$ is a probability measure. The function $\lambda \mapsto (1+\lambda)/(1-\lambda)$ is convex, so Jensen's inequality implies that the right hand side of (9.7.2) is bounded below by $\frac{1}{2}(1+\rho_h(1))/(1-\rho_h(1))$, where

$$\rho_h(1) = \frac{\int \lambda \, dE_h(\lambda)}{\int dE_h(\lambda)} = \frac{C_h(1)}{C_h(0)} = \frac{C_g(1)}{C_g(0)}.$$

This proves the proposition. □

9.7.2 Local algorithms

We shall begin by proving the two theorems about irreducibility of k-site algorithms from Section 9.4.1. Theorem 9.4.1 states that in two dimensions there are frozen $(6r+17)$-step walks for every $r \geq k$. Theorem 9.4.2 states that for $d = 2$ or 3 and for sufficiently large N, the cardinality of the largest ergodicity class of any k-site algorithm is less than $e^{-aN} c_N$ for some $a > 0$. We give the proof only for $d = 2$, as discussed in Section 9.4.1.

Proof of Theorem 9.4.1. Let $d = 2$. For each positive integer r, let $\psi^{(r)}$ be the $(6r+17)$-step walk

$$\mathrm{N}^r\mathrm{E}\mathrm{S}^{r+1}\mathrm{W}^2\mathrm{N}^{r+2}\mathrm{E}^5\mathrm{S}^{r+2}\mathrm{W}^2\mathrm{N}^{r+1}\mathrm{E}\mathrm{S}^r$$

(see Figure 9.5 in Section 9.4.1). We shall show that if $r \geq k$ then $\psi^{(r)}$ is frozen under k-step moves.

Let $N = 6r+17$. Let $\mathcal{B}$ be the set of all sites of $\psi^{(r)}$, so that $\mathcal{B}$ consists of an $(r+2) \times 6$ rectangle of sites of $\mathbf{Z}^2$. Consider removing any k contiguous sites $\psi^{(r)}(I), \ldots, \psi^{(r)}(I+k-1)$ from $\psi^{(r)}$. We want to find k distinct sites $a_1, \ldots, a_k$ such that: $|a_j - a_{j+1}| = 1$ for $j = 1, \ldots, k-1$; each a_j is in the set

$$\mathcal{D} \equiv (\mathbf{Z}^2 \setminus \mathcal{B}) \cup \{\psi^{(r)}(I), \ldots, \psi^{(r)}(I+k-1)\};$$

$|a_1 - \psi^{(r)}(I-1)| = 1$ if $I > 0$; and $|a_k - \psi^{(r)}(I+k)| = 1$ if $I < N-k+1$. If the only choice for each a_j is $\psi^{(r)}(I+j-1)$, then we can conclude that $\psi^{(r)}$ is indeed frozen.

If $I = 0$, then the removed sites all lie on a vertical line (since $r \geq k$). Moreover, the only nearest neighbour of $\psi^{(r)}(k)$ that is in $\mathcal{D}$ is $\psi^{(r)}(k-1)$, so we must take a_k to be this site. Similarly, the only choice for a_j is

$\psi^{(r)}(j-1)$ for each j, so no changes are possible when $I = 0$. Similarly, no changes are possible when $I = N - k + 1$.

Suppose now that $0 < I < N - k + 1$. The proof now is essentially by inspection. Look at all possibilities for $\psi^{(r)}(I-1)$ and $\psi^{(r)}(I+k)$ (in particular, whether or not they are on the boundary of the box $\mathcal{B}$), and look at how to connect these points by a $(k+1)$-step self-avoiding walk which only passes through points of $\mathcal{D}$. On the one hand, if at least one of these two points are on the boundary of $\mathcal{B}$, then there is only one possible walk (in fact, $k+1$ is the length of the shortest such walk that joins these points). On the other hand, if neither point is on the boundary of $\mathcal{B}$, then the only points of $\mathcal{D}$ that can be reached are precisely $\psi^{(r)}(I), \ldots, \psi^{(r)}(I+k-1)$, and there is no choice but to take them in their correct order (since they lie on either one or two vertical lines, and since all k of them must be used). □

Proof of Theorem 9.4.2. Let $d = 2$ and fix k. Let P be the $(10k+39)$-step pattern

$$\mathrm{N}^{k+2}\mathrm{W}^3\mathrm{S}^{k+1}\mathrm{E}\mathrm{N}^k\mathrm{E}\mathrm{S}^{k+1}\mathrm{W}^3\mathrm{N}^{k+3}\mathrm{E}^9\mathrm{S}^{k+3}\mathrm{W}^3\mathrm{N}^{k+1}\mathrm{E}\mathrm{S}^k\mathrm{E}\mathrm{N}^{k+1}\mathrm{W}^3\mathrm{S}^{k+2}$$

(see Figure 9.11), and let $L = 10k + 39$. An argument similar to the proof of Theorem 9.4.1 shows that if P occurs at the m-th step of a given self-avoiding walk ω, then P must occur at the m-th step of every walk that is in the same ergodicity class as ω.

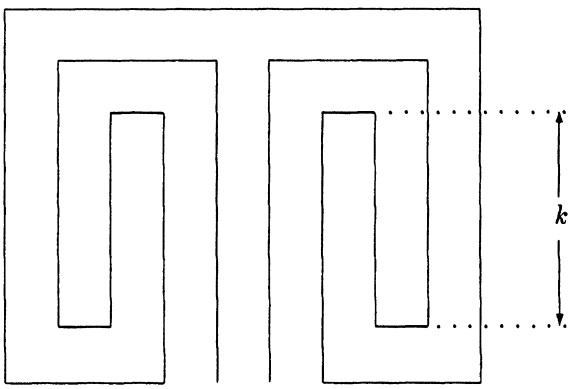

Figure 9.11: The pattern P from the proof of Theorem 9.4.2.

For every integer $t \geq 0$, and for every sequence $0 \leq m_1 < m_2 < \ldots < m_t < N$, let $\mathcal{E}_N(m_1, m_2, \ldots, m_t)$ denote the set of walks in $\mathcal{S}_N$ such that P

occurs at the m_j-th step of ω for every $j = 1, \ldots, t$ and nowhere else in ω. (For $t = 0$, this is the set of walks on which P does not occur.) Notice that successive occurrences of P in ω cannot overlap, and so we always have $m_j - m_{j-1} > L$ whenever $\mathcal{E}_N(m_1, m_2, \ldots, m_t)$ is nonempty. For each $t \geq 0$, let

$$M(t,k,N) = \max\{|\mathcal{E}_N(m_1, m_2, \ldots, m_t)| : 0 \leq m_1 < \ldots < m_t < N\}. \tag{9.7.3}$$

By the conclusion of the preceding paragraph, each ergodic class is contained in some $\mathcal{E}_N(m_1, m_2, \ldots, m_t)$, and so

$$CLEC_{k,N} \leq \max_{t \geq 0} M(t,k,N). \tag{9.7.4}$$

Since P is a proper internal pattern, Kesten's Pattern Theorem 7.2.3 tells us that there exists an $a > 0$ such that P must occur at least aN times on "almost all" N-step walks, i.e.

$$\limsup_{N \to \infty} (c_N[aN, P])^{1/N} < \mu. \tag{9.7.5}$$

Therefore

$$\limsup_{N \to \infty} \left[\max_{0 \leq t \leq aN} M(t,k,N) \right]^{1/N} < \mu. \tag{9.7.6}$$

Next, we claim that for any $t \geq 0$ and any sequence $0 \leq m_1 < \ldots < m_t < N$,

$$\mathcal{E}_N(m_1, m_2, \ldots, m_t) \leq c_{N-t(L-1)}. \tag{9.7.7}$$

To see this, define the function f from $\mathcal{E}_N(m_1, m_2, \ldots, m_t)$ to $\mathcal{S}_{N-t(L-1)}$ which removes each occurrence of P and replaces it by a single bond. Since f is one-to-one, the bound (9.7.7) follows. Therefore

$$\begin{aligned} \limsup_{N \to \infty} \left[\max_{t \geq aN} M(t,k,N) \right]^{1/N} &\leq \limsup_{N \to \infty} \left[\max_{t \geq aN} c_{N-t(L-1)} \right]^{1/N} \\ &= \mu^{1-a(L-1)} < \mu. \end{aligned} \tag{9.7.8}$$

The theorem now follows from (9.7.8) and (9.7.6). □

9.7.3 The pivot algorithm

We begin with a proof of the irreducibility of the pivot algorithm and some of its variants, which is asserted in Theorem 9.4.4. More precisely, this theorem says that any walk in $\mathcal{S}_N$ can be transformed into a straight walk

by a sequence of at most $2N-1$ pivots, each of which is either a reflection through a coordinate hyperplane or a rotation by $\pm\pi/2$.

Proof of Theorem 9.4.4. We begin with some notation. For each N-step self-avoiding walk ω and for each $j = 1, \ldots, d$, let

$$m_j^1(\omega) = \min\{\omega_j(k) : k = 0, 1, \ldots, N\} \tag{9.7.9}$$

and

$$m_j^2(\omega) = \max\{\omega_j(k) : k = 0, 1, \ldots, N\} \tag{9.7.10}$$

denote the minimum and maximum values of the j-th coordinate of the sites of ω, and let

$$M_j(\omega) = m_j^2(\omega) - m_j^1(\omega) \tag{9.7.11}$$

denote the extension of ω in the j-th coordinate direction. Let $\mathcal{B}(\omega)$ denote the smallest rectangular box containing ω, i.e.

$$\mathcal{B}(\omega) = \{x \in \mathbf{Z}^d : m_j^1(\omega) \le x \le m_j^2(\omega) \text{ for all } j = 1, \ldots, d\}, \tag{9.7.12}$$

and let

$$D(\omega) = M_1(\omega) + \cdots + M_d(\omega) \tag{9.7.13}$$

denote the l^1 diameter of $\mathcal{B}(\omega)$. A *face* of $\mathcal{B}(\omega)$ is any set of the form $\{x \in \mathcal{B}(\omega) : x_j = m_j^i(\omega)\}$ for some $i = 1, 2$ and some $j = 1, \ldots, d$. Finally, let

$$A(\omega) = |\{k : 0 < k < N \text{ and } \omega(k) = \frac{1}{2}[\omega(k-1) + \omega(k+1)]\}| \tag{9.7.14}$$

denote the number of straight internal angles of ω.

The strategy of the proof is the following. Observe that for every N-step self-avoiding walk ω, we have $0 \le D(\omega) \le N$ and $0 \le A(\omega) \le N-1$, and moreover $D(\omega)+A(\omega) = 2N-1$ if and only if ω is a straight walk. It suffices to show that if ω is not straight, then there exists another self-avoiding walk $\tilde{\omega}$ such that $D(\tilde{\omega}) + A(\tilde{\omega}) > D(\omega) + A(\omega)$ and $\tilde{\omega}$ can be obtained from ω by either a single reflection through a coordinate hyperplane or a single rotation by $\pm\pi/2$. Specifically, we shall show that if there is a face of $\mathcal{B}(\omega)$ which contains neither of the endpoints $\omega(0)$ nor $\omega(N)$, then a reflection through that face will increase D but not change A; and if no such face exists, then there exists a rotation that increases A by one without decreasing D.

We now give the details. Consider an arbitrary N-step self-avoiding walk ω that is not straight. We shall consider two cases separately. Since ω is fixed, we shall write $\mathcal{B}$ for $\mathcal{B}(\omega)$ and m_j^i for $m_j^i(\omega)$.

Case I. Suppose that there exists an $i \in \{1,2\}$ and a $j \in \{1,\ldots,d\}$ such that neither $\omega(0)$ nor $\omega(N)$ lies in the face $\{x \in \mathcal{B} : x_j = m_j^i\}$. Let t be the smallest index such that $\omega(t)$ lies in this face. Now let $\tilde{\omega}$ be the walk obtained by reflecting $\omega(t+1),\ldots,\omega(N)$ through the hyperplane $x_j = m_j^i$: that is, $\tilde{\omega}(k) = \omega(k)$ for each $k \le t$, while the coordinates of $\tilde{\omega}(k)$ for $k > t$ are given by

$$\tilde{\omega}_l(k) = \begin{cases} 2m_j^i - \omega_j(k) & \text{if } l = j \\ \omega_l(k) & \text{if } l \neq j. \end{cases} \tag{9.7.15}$$

(See Figure 9.12.) It is not hard to see that $\tilde{\omega}$ is self-avoiding, and that

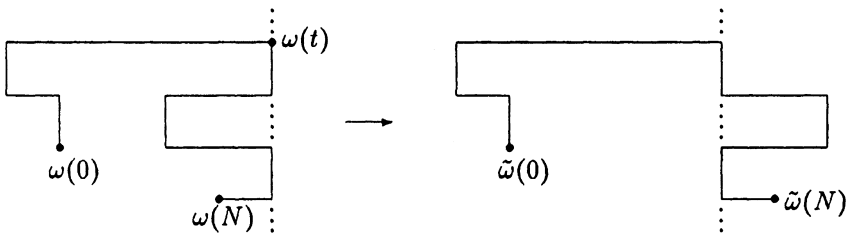

Figure 9.12: Case I of the proof of Theorem 9.4.4: reflection through the hyperplane denoted by the dotted line.

$A(\tilde{\omega}) = A(\omega)$ [notice that both $\tilde{\omega}$ and ω have right angles at $\omega(t)$]. Let us now show that $D(\tilde{\omega}) > D(\omega)$. Writing $M_j(\omega[r,s])$ to denote the extension of the subwalk $(\omega(r),\ldots,\omega(s))$ in the j-th coordinate direction, we see that

$$M_j(\omega) = \max\{M_j(\omega[0,t]), M_j(\omega[t,N])\} \tag{9.7.16}$$

and

$$M_j(\tilde{\omega}) = M_j(\omega[0,t]) + M_j(\omega[t,N]). \tag{9.7.17}$$

Since $\omega_j(0) \neq m_j^i$ and $\omega_j(N) \neq m_j^i$, both $M_j(\omega[0,t])$ and $M_j(\omega[t,N])$ are strictly positive, and so we conclude that $M_j(\tilde{\omega}) > M_j(\omega)$. Since $M_l(\tilde{\omega}) = M_l(\omega)$ whenever $l \neq k$, this proves that $D(\tilde{\omega}) > D(\omega)$, and hence that $D(\tilde{\omega}) + A(\tilde{\omega}) > D(\omega) + A(\omega)$. This completes the proof for Case I.

Case II. Suppose that ω is not covered by Case I; that is, suppose that every face of $\mathcal{B}$ contains at least one endpoint. This means that $\omega(0)$ and $\omega(N)$ are in diagonally opposite corners of the box $\mathcal{B}$. Since ω is not straight, let q be the largest index such that ω has a right angle at $\omega(q)$:

$$q = \max\{k : 0 < k < N \text{ and } \omega(k) \neq \frac{1}{2}[\omega(k-1) + \omega(k+1)]\}. \tag{9.7.18}$$

Since $\omega(N)$ is in a corner of $\mathcal{B}$, we will be able to perform a $\pm\pi/2$ rotation to straighten out the angle at $\omega(q)$. To be precise: the sites $\omega(q),\ldots,\omega(N)$ lie on a straight line perpendicular to the line segment joining $\omega(q-1)$ to $\omega(q)$. Let α be the coordinate such that $\omega_\alpha(q-1) \neq \omega_\alpha(q)$, and let β be the coordinate such that $\omega_\beta(q) \neq \omega_\beta(N)$. Observe that $\alpha \neq \beta$. Now we can define a new walk $\tilde{\omega}$ by choosing $\omega(q)$ as a pivot site and performing a rotation in the (x_α, x_β)-plane to get a straight angle at $\tilde{\omega}(q) = \omega(q)$. (See Figure 9.13.) The resulting walk has $\tilde{\omega}(q-1),\tilde{\omega}(q),\ldots,\tilde{\omega}(N)$ all

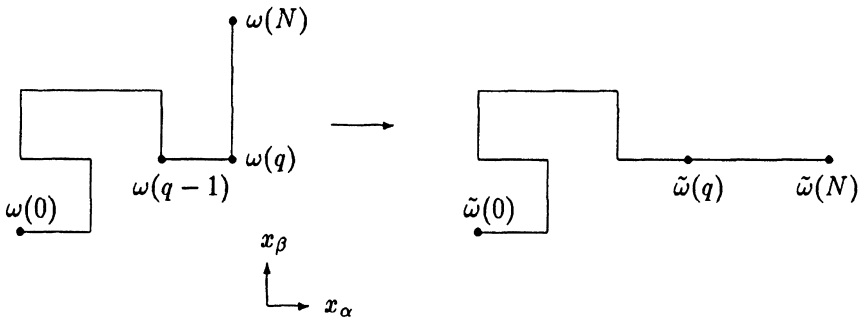

Figure 9.13: Case II of the proof of Theorem 9.4.4: rotation by $-\pi/2$. Also shown are the coordinate directions x_α and x_β.

on a straight line. Since $\tilde{\omega}(0),\ldots,\tilde{\omega}(q-1)$ are on the opposite side of the hyperplane $x_\alpha = \tilde{\omega}_\alpha(q)$ from $\tilde{\omega}(q+1),\ldots,\tilde{\omega}(N)$, we see that $\tilde{\omega}$ is self-avoiding. We also see that

$$M_\alpha(\tilde{\omega}) = M_\alpha(\omega) + N - q, \tag{9.7.19}$$

that

$$M_\beta(\tilde{\omega}) = M_\beta(\omega[0,q]) \geq M_\beta(\omega) - (N-q), \tag{9.7.20}$$

and that $M_j(\tilde{\omega}) = M_j(\omega)$ for all $j \neq \alpha, \beta$. Therefore $D(\tilde{\omega}) \geq D(\omega)$. Also, we clearly have $A(\tilde{\omega}) = A(\omega) + 1$, and hence that $D(\tilde{\omega}) + A(\tilde{\omega}) > D(\omega) + A(\omega)$. This completes the proof for Case II, and we are done. □

Next we consider the pivot algorithm applied to the ordinary random walk, without ever checking for intersections. To be precise, the state space of the algorithm is the set $\mathcal{S}_N^\circ$ of all $(2d)^N$ ordinary N-step walks starting at the origin, and the Generic Fixed-Length Dynamic Algorithm (from the beginning of Section 9.4) is implemented using the usual Step 2 for the pivot algorithm as described in Section 9.4.3, but in Step 3 $\omega^{[t+1]}$ is *always* set equal to $\tilde{\omega}$. For a given real-valued function g on $\mathcal{S}_N^\circ$, let $\tau_{exp,g}^\circ$ and $\tau_{int,g}^\circ$

respectively denote the exponential and integrated autocorrelation times of g with respect to this algorithm [as defined in (9.2.12) and (9.2.10)].

Proposition 9.7.1 *For each N, define the function $r^2 \equiv r_N^2$ on the set $\mathcal{S}_N^o$ by $r_N^2(\omega) = |\omega(N)|^2$, the squared end-to-end distance of $\omega \in \mathcal{S}_N^o$. Then as $N \to \infty$*

$$\tau_{exp,r^2}^o \sim N \tag{9.7.21}$$

and

$$\tau_{int,r^2}^o \sim 2 \log N. \tag{9.7.22}$$

Proof. Fix N. Using the notation $\Delta\omega^{[t]}(i) = \omega^{[t]}(i) - \omega^{[t]}(i-1)$ to denote the i-th step of the walk $\omega^{[t]}$, define $A_{ij}^{[t]}$ to be the dot product of the i-th and j-th steps of $\omega^{[t]}$:

$$A_{ij}^{[t]} = \Delta\omega^{[t]}(i) \cdot \Delta\omega^{[t]}(j)$$

for $1 \le i, j \le N$. Then by expanding the square we have

$$r^2(\omega^{[t]}) = \left| \sum_{i=1}^{N} \Delta\omega^{[t]}(i) \right|^2 = N + 2 \sum_{1 \le i < j \le N} A_{ij}^{[t]}. \tag{9.7.23}$$

In equilibrium, $\omega^{[t]}$ is uniformly distributed on $\mathcal{S}_N^o$, and so the N steps of $\omega^{[t]}$ are independent and uniformly distributed on the set of the $2d$ (positive and negative) unit vectors of $\mathbf{Z}^d$. By symmetry we have

$$E(A_{ij}^{[t]}) = 0 \quad \text{whenever } i \ne j, \tag{9.7.24}$$

and also

$$E[(A_{ij}^{[0]})^2] = \Pr\{|A_{ij}^{[0]}| = 1\} = \frac{1}{d}. \tag{9.7.25}$$

We also have that if $t \ge 0$, $i < j$, and $k < l$, then

$$E(A_{ij}^{[0]} A_{kl}^{[t]}) = 0 \quad \text{unless } i = k \text{ and } j = l. \tag{9.7.26}$$

Consider the first iteration of the pivot algorithm with initial walk $\omega^{[0]}$. For a given k between 1 and N, a necessary condition for the direction of the k-th step to change is that the chosen pivot site I is less than k. In fact, if G is the chosen symmetry, then

$$\Delta\omega^{[1]}(k) = \begin{cases} \Delta\omega^{[0]}(k) & \text{if } I \ge k \\ G\left(\Delta\omega^{[0]}(k)\right) & \text{if } I < k. \end{cases} \tag{9.7.27}$$

Therefore if I is not in the interval $[i,j)$, then $A_{ij}^{[1]} = A_{ij}^{[0]}$. Also, since G is chosen uniformly at random from $\mathcal{G}_d$, the vector $G\left(\Delta\omega^{[0]}(k)\right)$ is uniformly distributed on the set of unit vectors of $\mathbf{Z}^d$; moreover, it is *independent* of the entire walk $\omega^{[0]}$. In particular, we see that if I is in the interval $[i,j)$, then $A_{ij}^{[1]}$ is independent of $A_{ij}^{[0]}$.

Let $Q \equiv Q_{i,j,t}$ be the event that at least one of the pivot sites of the first t iterations is in the interval $[i,j)$. Then as in the preceding paragraph we see that conditioned on the occurrence of Q the quantities $A_{ij}^{[t]}$ and $A_{ij}^{[0]}$ are independent; hence by (9.7.24),

$$E(A_{ij}^{[0]} A_{ij}^{[t]} | Q) = 0. \tag{9.7.28}$$

If Q does not occur, then $A_{ij}^{[t]} = A_{ij}^{[0]}$, and hence by (9.7.25)

$$E(A_{ij}^{[0]} A_{ij}^{[t]} | Q^c) = E((A_{ij}^{[0]})^2 | Q^c) = E((A_{ij}^{[0]})^2) = \frac{1}{d}. \tag{9.7.29}$$

Since the probability of Q^c is $[1-(j-i)/N]^t$, we see from (9.7.28), (9.7.29) and (9.7.24) that

$$\mathrm{Cov}(A_{ij}^{[0]}, A_{ij}^{[t]}) = \frac{1}{d}\left(1 - \frac{j-i}{N}\right)^t \quad \text{for } i < j \text{ and } t \geq 0. \tag{9.7.30}$$

Using (9.7.23), (9.7.26) and (9.7.30), we see that for every $t \geq 0$

$$\begin{aligned} C_{r^2}(t) \equiv \mathrm{Cov}(r^2(\omega^{[0]}), r^2(\omega^{[t]})) &= 4 \sum_{1 \leq i < j \leq N} \mathrm{Cov}(A_{ij}^{[0]}, A_{ij}^{[t]}) \\ &= \frac{4}{d} \sum_{m=1}^{N-1} (N-m)\left(1 - \frac{m}{N}\right)^t \\ &= \frac{4N}{d} \sum_{m=1}^{N-1} \left(1 - \frac{m}{N}\right)^{t+1}. \end{aligned} \tag{9.7.31}$$

The $m = 1$ term in (9.7.31) is dominant; in fact,

$$\frac{4N}{d}\left(1 - \frac{1}{N}\right)^{t+1} \leq C_{r^2}(t) \leq \frac{4N(N-1)}{d}\left(1 - \frac{1}{N}\right)^{t+1},$$

and so the definition of exponential autocorrelation time in (9.2.12) implies that

$$\tau^o_{exp,r^2} = \frac{-1}{\log\left(1 - \frac{1}{N}\right)} = N + O(1), \tag{9.7.32}$$

which proves (9.7.21). Next, (9.7.31) tells us that $C_{r^2}(0) = 2N(N-1)/d$, and putting this and (9.7.31) into the definition of integrated autocorrelation time [recall (9.2.10)] yields

$$\begin{aligned} \tau^{o}_{int,r^2} &= \frac{1}{2} + \frac{d}{2N(N-1)} \sum_{t=1}^{\infty} \frac{4N}{d} \sum_{m=1}^{N-1} \left(1 - \frac{m}{N}\right)^{t+1} \\ &= \frac{1}{2} + \frac{2}{N-1} \sum_{m=1}^{N-1} \frac{\left(1 - \frac{m}{N}\right)}{\frac{m}{N}} \\ &= \frac{1}{2} + \frac{2}{N-1} \left[N \sum_{m=1}^{N-1} \frac{1}{m} - (N-1) \right] \\ &= 2 \log N + O(1), \end{aligned} \tag{9.7.33}$$

which proves (9.7.22). □

9.7.4 Fixed-endpoint methods

In this section we shall prove three results. Theorems 9.7.2 and 9.7.3 prove the irreducibility of the BFACF and DORS algorithms, respectively, in two dimensions. Finally, Theorem 9.7.4 proves that the exponential autocorrelation time of the BFACF algorithm is infinite.

We first establish some terminology that will be needed for the proofs of the first two theorems. Every self-avoiding polygon $\mathcal{P}$ in $\mathbf{Z}^2$ forms a simple closed curve, and hence has an inside and an outside (in the sense of the Jordan curve theorem). If v is a site of $\mathcal{P}$, then we say that v is *convex*, *concave*, or *straight* according as to whether the inside angle of $\mathcal{P}$ at v is 90°, 270°, or 180°.

Theorem 9.7.2 *For every nonzero endpoint x in $\mathbf{Z}^2$, the BFACF algorithm is irreducible on $\mathcal{S}(x)$.*

Proof. We begin by looking at a special case in which the endpoint is on the x_1-axis. For every integer $L > 0$, let $\rho^{(L)}$ denote the straight L-step walk from $(0,0)$ to $(L,0)$. For $N > L$, let $\mathcal{S}^*_N((L,0))$ be the set of N-step self-avoiding walks beginning at $(0,0)$ and ending at $(L,0)$ such that none of the sites $(1,0),(2,0),\ldots,(L-1,0)$ is occupied by a site of ω. We shall henceforth assume implicitly that N and L have the same parity, since otherwise $\mathcal{S}^*_N((L,0))$ is empty. Observe that $\mathcal{S}^*_N((L,0)) \subset \mathcal{S}_N((L,0))$ whenever $N > L > 0$, with equality if $L = 1$. Every walk ω in $\mathcal{S}^*_N((L,0))$ has an associated $(N+L)$-step self-avoiding polygon $\mathcal{P} \equiv \mathcal{P}(\omega)$ whose bonds are the N bonds of ω together with the L bonds of $\rho^{(L)}$.

Our first goal is to prove the following:

Claim A: Suppose that $N > L > 0$ and ω is in $\mathcal{S}_N^*((L,0))$. Then it is possible to transform ω into the straight walk $\rho^{(L)}$ by BFACF moves in such a way that none of the intermediate walks obtained in this process has a site lying outside of (the original) $\mathcal{P}(\omega)$.

Claim A implies irreducibility in the case $||x||_1 = 1$ (upon taking $L = 1$), and the approach that we take to prove it will also be used in the proof for general x.

To prove Claim A, we need some additional terminology. If ω is in $\mathcal{S}_N^*((L,0))$, then we say that the subwalk $\omega[i,j] \equiv (\omega(i),\ldots,\omega(j))$ $(0 \le i < j \le N)$ is a *U-turn* of ω if $j-i \ge 3$, $\omega[i+1, j-1]$ lies on a straight line, and the sites $\omega(i+1)$ and $\omega(j-1)$ are both convex sites of $\mathcal{P}(\omega)$. We say that $\omega(k)$ is an *obstruction* of the U-turn $\omega[i,j]$ if $\omega(k)$ is on the line segment whose endpoints are $\omega(i)$ and $\omega(j)$, and $k \ne i, j$. We say that the U-turn $\omega[i,j]$ is *unobstructed* if it has no obstructions. Observe that if $\omega[i,j]$ is

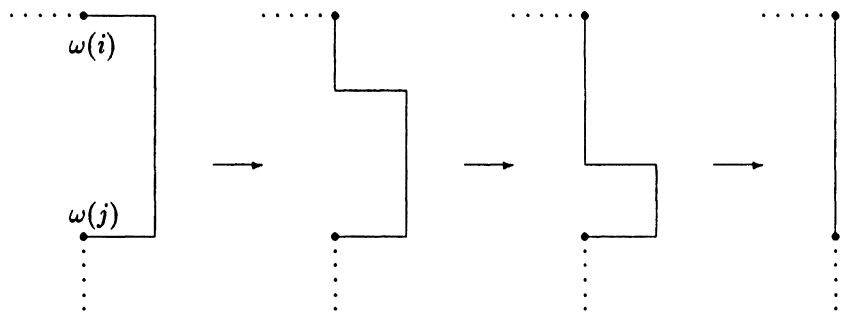

Figure 9.14: How the BFACF algorithm can use the presence of an unobstructed U-turn to shorten the length of a walk by 2.

an unobstructed U-turn of ω, then ω can be transformed into a walk ω' of length $N-2$ using $j-i-2$ BFACF moves as in Figure 9.14. Moreover, if $N-2 = L$ then $\omega' = \rho^{(L)}$, while if $N-2 > L$ then $\omega' \in \mathcal{S}_{N-2}^*((L,0))$. So Claim A will be proven if we can prove the following:

Claim B: For every $N > L > 0$, every walk in $\mathcal{S}_N^*((L,0))$ contains an unobstructed U-turn.

We now prove Claim B by induction on N.

Let $P(N)$ be the assertion that whenever L satisfies $N > L > 0$, every walk in $\mathcal{S}_N^*((L,0))$ contains an unobstructed U-turn. To start the induction, we note that $P(3)$ and $P(4)$ are clearly true. Let $N \ge 5$, and assume that

$P(n)$ is true for every $n < N$. Let ω be an arbitrary walk in $\mathcal{S}_N^*((L,0))$ for some L, with associated polygon $\mathcal{P}$. It is not hard to see that ω always contains a U-turn $\omega[i,j]$. (First observe that for every self-avoiding polygon, the number of convex sites exceeds the number of concave sites by exactly 4, because the sum of the signed inside angles must be exactly $+360°$. Therefore there must exist integers a and b with $0 < a < b < N$ such that $\omega(a)$ and $\omega(b)$ are both convex sites of $\mathcal{P}$ and such that if $b > a+1$ then the intervening sites $\omega(a+1), \ldots, \omega(b-1)$ are all straight sites of $\mathcal{P}$. Then $\omega[a-1, b+1]$ is a U-turn.) If it is unobstructed, then we are done, so assume that it has an obstruction $\omega(k)$. Then there exists an l satisfying $i+1 < l < j-1$ such that $||\omega(k) - \omega(l)||_1 = 1$. Suppose that $0 \le k < i$ (the same argument will work if $j < k \le N$). Let ζ denote the subwalk $\omega[k,l]$; since ζ has endpoints that are nearest neighbours, we can let $\mathcal{Q}$ denote its associated polygon. Observe that the bond $(\omega(k), \omega(l))$ lies inside $\mathcal{P}$, since we know that $\omega(i)$ and $\omega(j)$ are convex sites of $\mathcal{P}$. Therefore the inside of $\mathcal{Q}$ is a subset of the inside of $\mathcal{P}$, and hence all the sites of ω that are not part of ζ must lie outside of $\mathcal{Q}$. The inductive assumption tells us that ζ contains an unobstructed U-turn, and the observation of the preceding sentence guarantees that this must also be an unobstructed U-turn for ω. Therefore $P(N)$ is true.

We have now proven Claims B and A. To complete the proof of the theorem, consider the case of a general site x. Now the terms "inside" and "outside" are not meaningful, so we first need to find something to use in the place of U-turns. Let ω be an arbitrary walk in $\mathcal{S}(x)$, and let $N = |\omega|$. We say that the subwalk $\omega[i,j]$ $(0 \le i < j \le N)$ is a *C-turn* of ω if $j - i \ge 3$, $\omega[i+1, j-1]$ lies on a straight line that is perpendicular to the steps $\Delta\omega(i+1)$ and $\Delta\omega(j)$, and $\Delta\omega(i+1) = -\Delta\omega(j)$. (Observe that in the case $||x||_1 = 1$, every U-turn is a C-turn.) We define *obstruction* for C-turns exactly as we did for U-turns. The walk ω has no C-turns if and only if it has minimal length, i.e. $N = ||x||_1$, and it is easy to see that any minimal length walk can be transformed into any other by BFACF moves. So suppose $N > ||x||_1$; to prove the theorem, we need to show that it is always possible to reduce the length of ω using BFACF moves. Analogously to the case $||x||_1 = 1$, it suffices to prove that ω must have a C-turn with no obstructions.

Let $\omega[I,J]$ be a smallest C-turn of ω (i.e., satisfying $J - I \le j - i$ for every other C-turn $\omega[i,j]$). If $\omega[I,J]$ has no obstructions, then we are done, so assume $\omega[I,J]$ has one or more obstructions. It is not hard to see that one of these obstructions must be an endpoint of ω (because otherwise the obstructions would have to be part of a C-turn that is smaller than $\omega[I,J]$, which contradicts our choice of I and J). Without loss of generality, assume that $\omega(0)$ is an obstruction of $\omega[I,J]$ [the same argument

will work for $\omega(N)$]. Let M be the (unique) integer such that $I+1 < M < J-1$ and $||\omega(0)-\omega(M)||_1 = 1$. Let ζ denote the subwalk $\omega[0,M]$; since ζ has endpoints that are nearest neighbours, we can let $\mathcal{Q}$ denote its associated polygon. There are two cases that could occur: either the sites $\omega(M+1),\ldots,\omega(N)$ all lie outside $\mathcal{Q}$, or else they all lie inside $\mathcal{Q}$ (there are no other possibilities because the subwalks $\omega[0,M]$ and $\omega[M+1,N]$ cannot intersect). We shall consider these two cases in turn.

Case I: The sites $\omega(M+1),\ldots,\omega(N)$ all lie outside $\mathcal{Q}$. By Claim B, we see that ζ has an unobstructed U-turn; since the sites of ω that are not part of ζ all lie outside $\mathcal{Q}$, this must be an unobstructed C-turn of ω.

Case II: The sites $\omega(M+1),\ldots,\omega(N)$ all lie inside $\mathcal{Q}$. Let $u = \Delta\omega(N)$ be the direction of the last step of ω. Let

$$L = \min\{l > 0 : \omega(N) + lu \text{ is a site of } \omega\}$$

(note that $L < \infty$ because $\omega(N)$ lies inside $\mathcal{Q}$). Let t be the integer such that $\omega(t) = \omega(N)+Lu$. Observe that the subwalk $\omega[t,N]$ is (the translation and rotation/reflection of) a walk in $\mathcal{S}^*_{N-t}((L,0))$. Let $\mathcal{R}$ be the polygon consisting of the bonds of $\omega[t,N]$ and the straight line segment joining $\omega(N)$ to $\omega(t)$. Then the inside of $\mathcal{R}$ is a subset of the inside of $\mathcal{Q}$. In particular, the sites $\omega(0),\ldots,\omega(t-1)$ all lie outside $\mathcal{R}$. Now Claim B shows that the subwalk $\omega[t,N]$ has an unobstructed U-turn, and since the rest of ω lies outside $\mathcal{R}$, this must also be an unobstructed C-turn of the entire walk ω. This proves the theorem in Case II. □

Recall that the state space of the DORS algorithm is the set of equivalence classes of N-step self-avoiding polygons in $\mathbf{Z}^2$ (Definition 3.2.2). Recall also that these polygons are "unrooted", as opposed to the set of polygons associated with $\mathcal{S}_{N-1}(e)$ (where $||e||_1 = 1$) which are "rooted" by the bond $(0,e)$ which can never be moved.

Theorem 9.7.3 *For every even N, the DORS algorithm is irreducible for unrooted N-step polygons in two dimensions. In fact, if $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are N-step polygons in $\mathbf{Z}^2$, then there is a sequence of at most $2N-2$ transformations that transforms $\mathcal{Q}_1$ into $\mathcal{Q}_2$.*

Proof. Let $c(\mathcal{P})$ denote the total number of convex and concave sites on the polygon $\mathcal{P}$. A *rectangle* is a polygon $\mathcal{P}$ that has $c(\mathcal{P}) = 4$. The theorem is an immediate consequence of the following two facts.

A. Any polygon $\mathcal{P}$ that is not a rectangle can be transformed into some other polygon $\mathcal{Q}$ having $c(\mathcal{Q}) = c(\mathcal{P}) - 2$ using at most two transformations.

B. Any rectangle can be transformed into any other rectangle using one inversion and one reversing diagonal reflection.

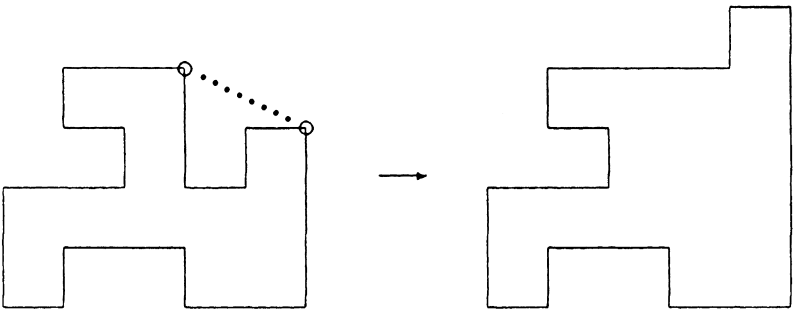

Figure 9.15: Proof of Theorem 9.7.3: a polygon with a supporting chord that is not parallel to a coordinate axis (left). Using the chord's endpoints as pivot sites for an inversion decreases the number of right angles by 2 (right).

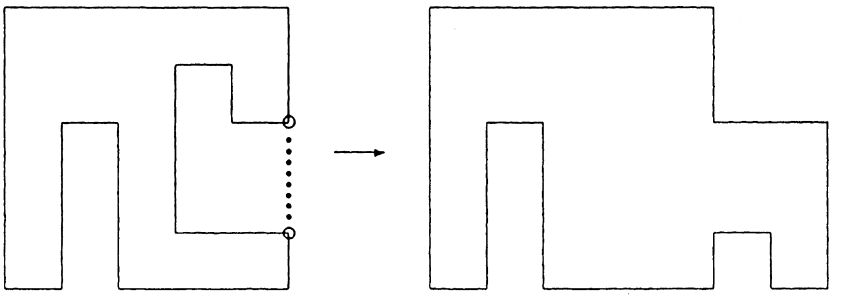

Figure 9.16: Proof of Theorem 9.7.3: a polygon whose two supporting chords are each parallel to a coordinate axis (left). Using one chord's endpoints as pivot sites for an inversion yields a polygon with a diagonal supporting chord (right).

We shall prove A first. A line segment is said to be a *supporting chord* of $\mathcal{P}$ if its endpoints are both on $\mathcal{P}$, its interior points are all on the outside of $\mathcal{P}$, and it is contained in the boundary of the convex hull of $\mathcal{P}$. (See Figure 9.15.) Observe that any polygon that is not a rectangle has a supporting chord.

Suppose that a polygon $\mathcal{P}$ has a supporting chord that is not parallel to either coordinate axis. It is not hard to see that performing an inversion on $\mathcal{P}$ with pivot sites chosen to be the endpoints of this supporting chord will yield a self-avoiding polygon $\mathcal{Q}$ with $c(\mathcal{Q}) = c(\mathcal{P}) - 2$ (see Figure 9.15). Next, suppose that $\mathcal{P}$ is not a rectangle but each of its supporting chords is parallel to a coordinate axis (see Figure 9.16). Performing an inversion on $\mathcal{P}$ with pivot sites chosen to be the endpoints of some supporting chord will yield a self-avoiding polygon $\mathcal{P}'$ with $c(\mathcal{P}') = c(\mathcal{P})$, and it is not hard to see that this $\mathcal{P}'$ will have a supporting chord that is not parallel to either coordinate axis. Thus we have proven A.

Finally we turn to fact B, whose proof is illustrated in Figure 9.17. Let

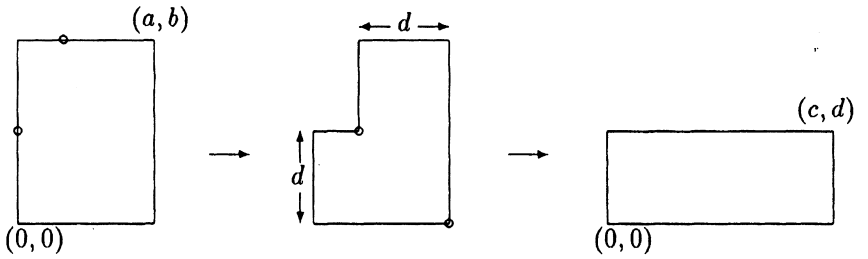

Figure 9.17: The DORS algorithm transforming one rectangle into another, using an inversion followed by a reversing diagonal reflection. The pivot sites are denoted by circles.

$\mathcal{R}_1$ and $\mathcal{R}_2$ be two N-step rectangles. Assume that the corners of $\mathcal{R}_1$ are at $(0,0)$, $(a,0)$, (a,b), and $(0,b)$, while the corners of $\mathcal{R}_2$ are at $(0,0)$, $(c,0)$, (c,d), and $(0,d)$, where a, b, c, and d are all positive and $a+b = c+d$. Without loss of generality, we can assume that $c > a \geq d$. Performing an inversion on $\mathcal{R}_1$ with pivot sites $(0,d)$ and $(a-d,b)$ gives a polygon $\mathcal{U}$, which in turn can be transformed into $\mathcal{R}_2$ by performing a reversing diagonal reflection with pivot sites $(a,0)$ and $(a-d,d)$. □

Theorem 9.7.4 *The exponential autocorrelation time τ_{exp} for the BFACF algorithm is infinite (for every x and z).*

Proof. Fix x and z. Let ϕ and ψ be two points (walks) in the state space, and let $T[\phi, \psi]$ be the smallest value of n such that $P^n(\phi, \psi) > 0$; i.e. $T[\phi, \psi]$ is the smallest time in which it is possible to get from ϕ to ψ. Let I_ϕ and I_ψ be the indicator functions of the singletons $\{\phi\}$ and $\{\psi\}$. Consider any $k < T[\phi, \psi]$. On the one hand, using the inner product defined in (9.2.14),

$$(I_\phi, (P^k - \Pi)I_\psi) = P^k(\phi, \psi)\pi(\phi) - \pi(\phi)\pi(\psi) = -\pi(\phi)\pi(\psi).$$

On the other hand, using (9.2.23) and (9.2.24), we have

$$|(I_\phi, (P^k - \Pi)I_\psi)| \le ||I_\phi||_2 ||P^k - \Pi||\, ||I_\psi||_2 = (\pi(\phi)\pi(\psi))^{1/2} \exp[-k/\tau_{exp}].$$

Combining the above two observations, rearranging and taking $k = T[\phi, \psi] - 1$ gives

$$\tau_{exp} \ge \frac{2(T[\phi, \psi] - 1)}{-\log(\pi(\phi)\pi(\psi))}. \tag{9.7.34}$$

This inequality says that if there are two states that are far apart, but not too unlikely, then τ_{exp} must be large.

To apply (9.7.34), consider $N >> ||x||_1$. Let ϕ be a shortest walk from 0 to x, and let $\psi = \psi^{[N]}$ be a walk of length N which does not intersect ϕ and whose shape is approximately square; this means that the area of the smallest surface whose boundary is the union of ϕ and $\psi^{[N]}$ is approximately $N^2/16$. Since the BFACF algorithm only modifies a walk by adding and removing bonds around a single lattice square, this surface area cannot change by more than 1 in a single iteration. Therefore $T[\phi, \psi^{[N]}] \simeq N^2$. Also, $\pi(\phi) = ||x||_1 z^{||x||_1}/\Xi(z, x)$ and $\pi(\psi^{[N]}) = Nz^N/\Xi(z, x)$, so the right side of (9.7.34) behaves like a constant times N as $N \to \infty$. Therefore, since N can be arbitrarily large, we must have $\tau_{exp} = +\infty$. □

9.8 Notes

Section 9.1. One of the best general overviews of Monte Carlo methods is Hammersley and Handscomb (1964), whose age has done remarkably little to diminish its appeal. Bratley, Fox and Schrage (1987) is a more recent general reference to various theoretical and practical issues in simulation and Monte Carlo. Binder and Heermann (1988) is a useful step-by-step guide to the practical aspects of Monte Carlo experiments in statistical mechanics. Kremer and Binder (1988) is a detailed survey of Monte Carlo methods for polymers in general. Sokal (1991) is a review on the problem of critical slowing-down.

General references for the theory and applications of Markov chains include Feller (1968), Karlin and Taylor (1975), and Nummelin (1984).

These authors and others usually say that a Markov chain is *ergodic* if it is irreducible, positive recurrent, and aperiodic. Most chains arising in Monte Carlo are positive recurrent and aperiodic, and for these chains questions of ergodicity are equivalent to questions of irreducibility. Although the Monte Carlo literature tends to use the term "ergodicity" when discussing irreducibility, we prefer the term "irreducibility" in this book to emphasize the specific nature of these problems.

The classic reference for hash tables is Knuth (1973), which is still highly recommended. Hashing is also treated in most computer science books on data structures. Early uses of hash tables for the self-avoiding walk problem are Gans (1965) and Jurs and Reissner (1971); a description is also given in Madras and Sokal (1988).

Section 9.2. Two general references on statistics are Silvey (1970) and Cox and Hinkley (1974). References on time series analysis include Priestley (1981) and Brockwell and Davis (1987). Bratley, Fox, and Schrage (1987) discuss time series analysis and other statistical issues in the specific context of simulation. Geyer (1992) and Gelman and Rubin (1992) present two contrasting views on problems of statistical inference for Markov chain simulations.

Proposition 9.2.2, Corollary 9.2.3, and Proposition 9.2.4 are from Appendix A of Caracciolo, Pelissetto, and Sokal (1990). The proof of Equation (9.2.21) in Section 9.7.1 is from Sokal and Thomas (1989), who actually prove a stronger theorem. The proof of Equation (9.2.27) is from Sokal (1989). The exposition of Section 9.2.3 is largely based upon the preceding three papers.

Section 9.3. Strides and biased sampling are reviewed in Hammersley and Handscomb (1964). Kremer and Binder (1988) includes a more recent review of biased sampling, with many references. The dimerization method is due to Suzuki (1968) and Alexandrowicz (1969). The derivation of (9.3.3) is from Madras and Sokal (1988).

Section 9.4. Many local algorithms have appeared in the literature; see Madras and Sokal (1987) and Kremer and Binder (1988) for some references. The failure of irreducibility for local algorithms was noticed early, by Heilmann (1968) (who observed that knots could cause problems) and by Verdier (1969) (who noted the existence of three-dimensional frozen configurations analogous to Figure 9.2). Theorems 9.4.1 and 9.4.2, as well as Proposition 9.4.3, are due to Madras and Sokal (1987); as explained there, the methods also allow one to prove Theorem 9.4.1 in $d = 3$. The proof of (9.4.1) under the stated assumption is due to Caracciolo *et al.* (1990).

Wall and Mandel (1975) commented that the probability of frozen configurations for the slithering snake algorithm did not tend to 0, but expected it to be negligibly small for practical purposes. The rigorous proof of the former assertion [Equation (9.4.2)] is due to Madras (1988).

Reiter (1990) proved irreducibility for a fixed-length algorithm in the spirit of the slithering snake: in this algorithm, a single bond in the walk can be replaced by a 3-bond U while simultaneously removing two bonds from the ends (and of course the reverse of this move can also be done).

The pivot algorithm has been independently rediscovered by many different authors since Lal (1969): Curro (1974), Olaj and Pelinka (1976), and MacDonald *et al.* (1985). Continuum analogues have been used by Stellman and Gans (1972) and Freire and Horta (1976). Except where cited otherwise, the results and discussion of Section 9.4.3 are from Madras and Sokal (1988).

Section 9.5. The rigorous proof of (9.5.3) appeared in Caracciolo *et al.* (1990).

Section 9.6. The BFACF algorithm is due to Berg and Foerster (1981), Aragão de Carvalho, Caracciolo and Fröhlich (1983), and Aragão de Carvalho and Caracciolo (1983); some ambiguities in these papers about the details of the algorithm were clarified in Caracciolo *et al.* (1990), whose presentation we follow here. The irreducibility of the BFACF algorithm in two dimensions (Theorem 9.7.2) is due to Madras (1986, unpublished). The bound (9.6.11) is due to Caracciolo *et al.* (1990).

Janse van Rensburg, Whittington, and Madras (1990) described a nonlocal fixed-length algorithm for polygons on the face-centred cubic lattice, and proved that it is irreducible.