Monte Carlo methods for the self-avoiding walk

2009 J. Phys. A: Math. Theor. 42 323001

(http://iopscience.iop.org/1751-8121/42/32/323001)

J. Phys. A: Math. Theor. **42** (2009) 323001 (97pp) doi:10.1088/1751-8113/42/32/323001

TOPICAL REVIEW

Monte Carlo methods for the self-avoiding walk

E J Janse van Rensburg

Department of Mathematics and Statistics, York University, Toronto, ON M3J 1P3, Canada

E-mail: rensburg@yorku.ca

Received 6 April 2009, in final form 5 June 2009
Published 22 July 2009
Online at stacks.iop.org/JPhysA/42/323001

Abstract

The numerical simulation of self-avoiding walks remains a significant component in the study of random objects in lattices. In this review, I give a comprehensive overview of the current state of Monte Carlo simulations of models of self-avoiding walks. The self-avoiding walk model is revisited, and the motivations for Monte Carlo simulations of this model are discussed. Efficient sampling of self-avoiding walks remains an elusive objective, but significant progress has been made over the last three decades. The model still poses challenging numerical questions however, and I review specific Monte Carlo methods for improved sampling including general Monte Carlo techniques such as Metropolis sampling, umbrella sampling and multiple Markov Chain sampling. In addition, specific static and dynamic algorithms for walks are presented, and I give an overview of recent innovations in this field, including algorithms such as flatPERM, flatGARM and flatGAS.

PACS numbers: 02.50.Ng, 02.70.Uu, 05.10.Ln, 36.20.Ey, 61.41.+e, 64.60.De, 89.75.Da

Contents

1. Introduction

The simulation and enumeration of polymer conformations remains one of the most fundamental problems in polymer physics (see, for example [27, 39]). Polymer statistics and enumeration poses a set of challenging questions in areas as diverse as enumerative combinatorics, statistics and statistical mechanics. In this review, I shall focus mainly on the Monte Carlo simulation of polymer conformations as modelled by the self-avoiding walk, which is the simplest model of a linear polymer in a good solvent.

The non-Markovian character of the self-avoiding walk is the source of many of the difficult and unresolved mathematical problems it poses. The model is also related to certain lattice field theories where it appears as a summation over (discrete) lattice paths, see for example [143].

Self-avoiding walks (see figure 1) have been studied since the 1940s as the most basic model of a linear polymer [38]. Significant theoretical progress in the mathematical description of the model is due to applications of probability theory, rigorous constructive techniques, scaling arguments and conformal field theory (in two-dimensional self-avoiding walk models).

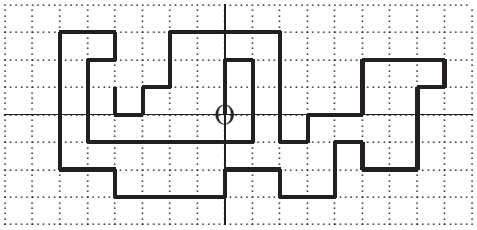

Figure 1. A self-avoiding walk in the square lattice. The walk starts at the origin and steps on distinct lattice sites. The final vertex of the walk is trapped, since the walk cannot be continued by adding another edge to its endpoint.

The field has advanced in major ways over the last 50 years, but the most basic question (the determination of the number of walks of given length) remains unresolved.

The generation of exact series for lattice walks has produced remarkably accurate results for critical exponents and connective constants [59, 61]. This approach has proven to be a superior method for obtaining numerical data on walks, and recent advances have extended series for walks to remarkable lengths in the square lattice [51–53, 82, 83, 148] and in the cubic lattice [19] with the result that Monte Carlo simulations have lagged considerably in accuracy when used to verify series results.

The results of series analysis should be considered against the backdrop that exact (but non-rigorous) numerical values for certain scaling exponents of the self-avoiding walk have been obtained in the square lattice by Coulomb gas [114] and conformal field theory [34] methods. The series data have verified many of these exact values to remarkable accuracy and perhaps even to a degree that cannot be obtained by Monte Carlo simulations as a matter of principle. This may be so even in the cubic lattice, where series analysis (using a lace expansion technique) has been used to provide good to excellent estimates for three-dimensional self-avoiding walk exponents and the cubic lattice connective constant [19]. While Monte Carlo may not be able to compete with these results, there is nevertheless the additional motivation for the use of Monte Carlo simulations to verify the results obtained by analysing series data.

Monte Carlo simulations of self-avoiding walks are a collection of versatile and robust algorithms. Many of these algorithms can be very generally applied to more general models of walks in confined spaces or interacting walks as models of interacting polymers. Generally, one should consider the Monte Carlo simulations of self-avoiding walks to be a method of last resort, to be used primarily when no other method would do, or when there is the need for the verification of (already known) results obtained by other means.

There are extensive results on the Monte Carlo simulations of interacting models of self-avoiding walks in the physics literature. These methods have been used to simulate models of polymers in dilute solution undergoing adsorption or collapse, amongst many other phenomena [25, 26]. The use of classical scaling arguments [27] together with numerical approaches, have had measured successes in describing these models [139, 149]. From a numerical point of view these models are difficult and normally require the use of efficient algorithms to estimate critical exponents and phase boundaries; see for example [44, 77].

The motivation for simulating lattice self-avoiding walks may originally have been found in the polymer enumeration problem, but the field has matured since the first simulations in the 1950s [132] to include not only a variety of algorithms and results, but also to pose its own independent questions about the nature of the available algorithms. These questions concern the efficacy of a given algorithm, including its ergodicity properties, its efficiency,

the appropriate application of the algorithm, and so on. Some of these questions have been addressed to various degrees in the literature; for example, see the review in [139] and chapter 9 in [100].

The invention of new Monte Carlo algorithms for sampling self-avoiding walks is also a topic of independent research interest [139], and numerous ingenious methods have been designed since the invention of the Rosenbluth method in 1955 [132]. Determining the properties of a given method is itself a mathematical issue, as it frequently poses non-trivial questions involving discrete time stochastic processes.

Generally, a numerical approach to the self-avoiding walk may be motivated by the need to verify results obtained by other means, or to determine the numerical value of an exponent or other constant associated with the model. If a Monte Carlo technique is used, then several questions arise, such as (1) how to sample walks efficiently from a given (statistical) ensemble, using appropriate data structures, (2) how to collect and measure data on the sampled walks efficiently and (3) how to analyse data acquired by the numerical method. Each of these questions has merit in its own right, and the efficiency of the simulation requires that a satisfactory approach exists for each.

The Monte Carlo method for statistical sampling from a distribution was invented in 1949 [109], and its implementation via the Metropolis algorithm was demonstrated already in 1953 [110]. Efficient Monte Carlo simulations of self-avoiding walks (as models of polymers) started in 1955 with the invention of the Rosenbluth algorithm [132]. Subsequently there have been significant progress; including new algorithms as well as several 'dynamic algorithms' which have been used to simulate the dynamics of lattice polymer chains [149] (rather than static properties). Many of these 'Monte Carlo dynamics' approaches have, on closer examination, been found to be flawed in their application, but have inspired several new approaches and algorithms since the 1980s, including the BFACF algorithm [4, 8] and the pivot algorithm [92, 102]. The Rosenbluth method, on the other hand, has been generalized to PERM [44] and to GARM [131].

The identification of the self-avoiding walk in a half-space as a stochastic Loewner evolution with parameter $\kappa = 8/3$ (SLE-8/3) [93, 135] raises the new possibility of computing properties of self-avoiding walks by simulating SLE-8/3. Steps along these lines have been taken [87, 88], but while this is an exciting development, in this review I shall focus on the direct Monte Carlo simulation of walks in the hypercubic lattice.

The calculation of scaling exponents remains a significant motivating factor for the Monte Carlo simulation of walks, and in section 2 an overview of the basic scaling ideas in models of walks is given. Interacting models of self-avoiding walks are briefly reviewed as well, including collapsing walks, adsorbing walks and stretched and pulled walks.

In section 3, the notion of a self-avoiding walk atmosphere [130] is introduced. Several definitions are presented, and a short detour to an interacting model of atmospheric collapse in walks is made. Particular definitions of walk atmospheres are important in constructing Monte Carlo type dynamics on the state space of walks, and the discussion of algorithms will be predicated on particular properties of atmospheric statistics.

The relationship between atmospheric statistics in the self-avoiding walk, and Monte Carlo algorithms for walks, is explored in section 4, before the general outline of static and dynamic Monte Carlo algorithms are presented. The Metropolis algorithm [110], umbrella sampling [147] and multiple Markov chain Monte Carlo [41] are reviewed in the context of the implementation of dynamic Monte Carlo sampling along a Markov chain. In addition, the analysis of data and calculation of autocorrelation times for dynamic Monte Carlo algorithms are reviewed.

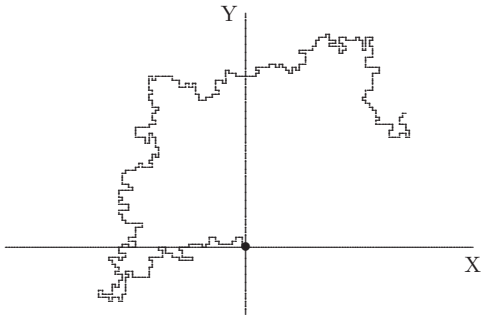

Figure 2. Self-avoiding walks of fixed length can be sampled uniformly along a Markov chain by using the pivot algorithm [102]. This walk of length 500 edges is obtained in a Markov chain realized by the pivot algorithm.

In sections 5–14, we review self-avoiding walk algorithms. Simple sampling in section 5 is technically the worst possible technique, and it is included as a baseline for comparison with better algorithms. The Rosenbluth method in section 6 is far more efficient, but is clearly based on the sampling technique of simple sampling. Dimerization is briefly reviewed in section 7.

Further generalizations of Rosenbluth-style sampling include the scanning method (section 8), PERM (section 9), GARM (section 10) and GAS (section 11). These are all static Monte Carlo algorithms based on simple sampling and the Rosenbluth algorithm, but with ingenious additions to improve sampling, such as pruning and enrichment (see [153]) of states in PERM [44].

In section 12, the pivot algorithm is presented. This is a dynamic Monte Carlo algorithm for sampling walks uniformly in the canonical (fixed length) ensemble. Grand canonical sampling of walks from a Boltzmann distribution over lengths is introduced by the Berretti–Sokal algorithm in section 13 or the BFACF algorithm (section 14).

In section 15, the application of Monte Carlo algorithms to lattice polygons is presented. The application of GARM and GAS algorithms, and the pivot and BFACF algorithms are reviewed.

The review is concluded in section 16 with a few observations and comments.

2. Self-avoiding walks

A *lattice self-avoiding walk* is a collection of distinct *vertices* $\{v_0, v_1, v_2, \ldots, v_n\}$ in a lattice together with a set of *edges* which are pairs of vertices of the form $\{(v_{i-1}, v_i)\}_{i=1}^{n}$, where $v_i \neq v_j$ if $i \neq j$, and the vertices in each edge are nearest neighbour lattice vertices. The Cartesian coordinates of a vertex v_i in the walk are denoted by $(X(v_i), Y(v_i), \ldots, Z(v_i))$ in d-dimensions with $X(v_i)$ the first coordinate, $Y(v_i)$ the second coordinate and $Z(v_i)$ the dth coordinate. See figures 1 and 2 for examples of self-avoiding walks in the square lattice.

Normally, the vertex v_0 is at the origin in the lattice, and this induces an orientation along the walk. The *length* of the walk is the number of edges it contains. We shall only consider walks in the d-dimensional hypercubic lattice $\mathbb{Z}^d$, where $d = 2$ or $d = 3$ in most cases.

The most fundamental quantity in the study of the self-avoiding walk is c_n, the number of walks of length n starting in the origin. For small values of $n, c_0 = 1, c_1 = 2d$,

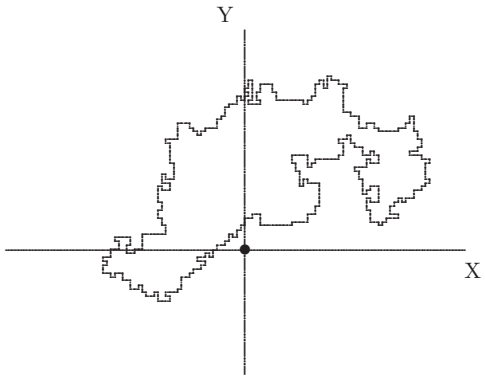

Figure 3. Self-avoiding polygons of fixed length can be sampled uniformly along a Markov chain by using the pivot algorithm [99]. This polygon of length 500 edges is obtained in a Markov chain realized by the pivot algorithm.

$c_2 = 2d(2d-1)$, and so on, but it quickly becomes very difficult to determine c_n. It is the case that

$$d^n \leqslant c_n \leqslant 2d(2d-1)^{n-1}, \tag{1}$$

so that c_n grows exponentially with n. The number c_n is known for $n \leqslant 71$ in $\mathbb{Z}^2$ [83] and for $n \leqslant 30$ in $\mathbb{Z}^3$ [19], and is difficult to determine.

If a self-avoiding walk of length $n+m$ is cut at its nth vertex, then two subwalks of lengths n and m are obtained. The number of choices of a walk of length $n+m$ is c_{n+m}, but the number of resulting subwalks is at most $c_n c_m$. Thus, c_n is a submultiplicative function on $\mathbb{N}$: $c_{n+m} \leqslant c_n c_m$. Together with the bounds in equation (1), this implies that the limit

$$\mu = \lim_{n\to\infty} [c_n]^{1/n} \tag{2}$$

exists and $d \leqslant \mu \leqslant (2d-1)$ [56, 58, 59].

Determining the growth constant μ is one of the most basic reasons for developing numerical algorithms for walks. In low dimensions, exact enumeration of walks to determine c_n has given the best known estimates for μ in $\mathbb{Z}^2$ [83] and $\mathbb{Z}^3$ [19] from self-avoiding walk data.

Kesten's pattern theorem [89, 90] shows that the limit

$$\lim_{n\to\infty} \frac{c_{n+2}}{c_n} = \mu^2 \tag{3}$$

exists, but it is not known that the limit $\lim_{n\to\infty}[c_{n+1}/c_n] = \mu$ exists. Is it known that $c_n \leqslant c_{n+1}$ [117], but there are few other proven properties of c_n. Scaling arguments [27] provide a general framework for examining c_n and more general models of interacting walks.

Closely related to walks are polygons: these are self-avoiding walks which return to the origin; as illustrated in figure 3. Polygons have a root vertex at the origin (this is the starting and final vertices in the walk), but normally polygons are counted up to equivalences under translations in the lattice. This removes the root, and p_n is defined as the number of (unrooted) polygons of length n steps in the hypercubic lattice. For example, $p_4 = 1$, $p_6 = 2$, $p_8 = 7$ in the square lattice, while $p_{2n+1} = 0$ in bipartite lattices such as $\mathbb{Z}^d$.

As for walks, p_n has been enumerated by computer to an astonishing $n = 110$ in the square lattice [85]. It is a theorem [57] that the limit

$$\mu = \lim_{n\to\infty} [p_{2n}]^{1/2n} \tag{4}$$

exists (it is taken through even integers in bipartite lattices such as $\mathbb{Z}^d$). The value of the limit is equal to μ; the growth constant of walks which is defined in equation (2) (see [57]).

It is known that the limit

$$\lim_{n\to\infty} \frac{p_{n+2}}{p_n} = \mu^2 \tag{5}$$

exists, a result which is due to Kesten [89, 90]; see [100] for a simpler proof.

2.1. Numerical estimates of μ

The numerical value of μ has been estimated in the square and cubic lattices using a variety of different methods. Computing it to high precision remains a motivating challenge for the development of numerical methods for the self-avoiding walk and for lattice polygons. Exact enumeration and series analysis of walks (and polygons) as well as Monte Carlo simulations have traditionally been used to estimate μ.

Series analysis for polygons [84, 85] gives μ in two dimensions to very high precision:

$$\mu = 2.638\,158\,530\,34 \pm 0.000\,000\,000\,10. \tag{6}$$

Jensen [82] can be consulted for a slight improvement on this estimate.

Determining μ from self-avoiding walk data is not nearly this accurate. The best estimate for μ obtained from self-avoiding walk data is

$$\mu = 2.638\,158\,56 \pm 0.000\,000\,03 \tag{7}$$

as determined in [83], see [54] for additional results.

Monte Carlo estimates for μ have been made using grand canonical Monte Carlo algorithms which sample self-avoiding walks from a distribution over their lengths. The most well-known such algorithm is the Beretti–Sokal algorithm [9]. In [112], it is reported that

$$\mu = 2.638\,164 \pm 0.000\,014, \tag{8}$$

based on simulations using the Berretti–Sokal algorithm. The error bar is a combined 95% statistical confidence interval and an estimated systematic error due to uncertainties in the model. A more generalized implementation of the Berretti–Sokal algorithm can be found in [113].

Canonical Monte Carlo simulations of self-avoiding walks (of fixed length) can be used to determine μ as well. Results reported in [130] show that

$$\mu = 2.638\,16 \pm 0.000\,12, \tag{9}$$

where an atmospheric statistic was used to estimate μ. A more recent simulation using an atmospheric statistic for polygons gave the estimate

$$\mu = 2.638\,05 \pm 0.000\,24, \tag{10}$$

see [78]. While these estimates do not compare well with the estimates based on exact enumeration data, they do serve the purpose of verifying digits in the best estimates in equations (6) and (7).

Less precise estimates for μ are available in three dimensions. Clisby *et al* [19] estimated that

$$\mu = 4.684\,043 \pm 0.000\,012, \tag{11}$$

by collecting series data on the cubic lattice self-avoiding walk using the lace expansion.

Collecting atmospheric statistics on polygons with canonical Monte Carlo simulations showed that

$$\mu = 4.683\,98 \pm 0.000\,16 \tag{12}$$

in three dimensions [78]. Hara *et al* [64] reported an (unpublished) estimate $\mu = 4.683\,907 \pm 0.000\,022$ due to Guttmann.

2.2. Scaling and critical exponents in self-avoiding walks

Scaling arguments presume that c_n and other mean observables computed for walks satisfy scaling laws which involve certain critical exponents. The limit in equation (2) shows that $c_n = \mu^{n+o(n)}$. A scaling assumption for c_n introduces a power-law correction to the expected exponential term above so that

$$c_n \sim A\mu^n n^{\gamma-1}. \tag{13}$$

The *entropic exponent* γ has mean field value $\gamma = 1$ (this is also the random walk value, since the number of random walks of length n is $(2d)^n$), but $\gamma > 1$ in two and three dimensions. In two dimensions conformal field theory gives the exact value $\gamma = 43/32$ [29], while numerical simulations have been used to estimate that $\gamma = 1.1575(6)$ in three dimensions [97]. The mean field value $\gamma = 1$ is the exact value in five and higher dimensions [62, 63], and also in four dimensions which is the upper critical dimension for self-avoiding walks, and where a logarithmic correction modifies equation (13) to $c_n \sim A\mu^n[\log n]^{1/4}$.

The generating function of c_n is called the *susceptibility* of the self-avoiding walk, and it is defined by

$$\chi(t) = \sum_{n=0}^{\infty} c_n t^n. \tag{14}$$

The radius of convergence of $\chi(t)$ is $t_c = \mu^{-1}$, and substitution of c_n using the right-hand side of equation (13) shows that

$$\chi(t) \sim \frac{A'}{(1-\mu t)^{\gamma}}. \tag{15}$$

Since $\gamma > 0$, the susceptibility is divergent as $t \nearrow \mu^{-1}$.

The mean of an observable $\mathcal{O}$ over all walks of length n is defined by

$$\langle \mathcal{O} \rangle_n = \frac{1}{c_n} \sum_{|\omega|=n} \mathcal{O}(\omega), \tag{16}$$

where the summation is over all walks of length n counted by c_n.

2.2.1. Metric scaling of walks.

Metric quantities of a self-avoiding walk s with vertices $\{v_0, v_1, \ldots, v_n\}$ are observables which have units [length]. These include the *span*

$$S(s) = \max_{i,j}\{|X(v_i) - X(v_j)|, |Y(v_i) - Y(v_j)|, \ldots, |Z(v_i) - Z(v_j)|\} \tag{17}$$

or the *average span*

$$S_a(s) = \frac{1}{d}\Big(\max_{i,j}\{|X(v_i) - X(v_j)|\} + \cdots + \max_{i,j}\{|Z(v_i) - Z(v_j)|\}\Big). \tag{18}$$

The mean square radius of gyration is also a metric quantity defined by

$$R^2(s) = \frac{1}{|s|^2} \sum_{i,j} \|v_i - v_j\|_2^2, \tag{19}$$

where $\|v\|_2$ is the Euclidean norm of the vector v defined by $\|v\|_2^2 = [X(v)]^2 + [Y(v)]^2 + \cdots + [Z(v)]^2$ for a given vector v.

The *mean span* $S_n \equiv \langle S(s)\rangle_n$ and *mean square radius of gyration* $R_n^2 \equiv \langle R^2(s)\rangle_n$ are defined by taking the mean over all walks of length n: since $n^{1/d} \leqslant S_n \leqslant n$ and $n^{2/d} \leqslant R_n^2 \leqslant n^2$, these metric quantities should scale as power laws with n:

$$R_n^2 \sim C_R\, n^{2\nu} \qquad \text{and} \qquad S_n \sim C_S\, n^{\nu}, \tag{20}$$

where ν is the *metric exponent*. The mean average span $\langle S_a(s)\rangle_n$ of walks of length n similarly scales as $\langle S_a(s)\rangle_n \sim C_A n^{\nu}$.

It is possible to define other metric quantities, including the volume of the smallest box containing the walk, the area of the image of the walk projected to lower dimensions and so forth. All these quantities have dimensions length raised to an integer power, and they scale as a power law with the length of the walk. Two other metric quantities which have received attention in numerical simulations are $\langle R_e^2\rangle_n$, the mean square end-to-end distance between the endpoints of a self-avoiding walk of length n, and $\langle R_m^2\rangle_n$, the mean square distance between a vertex in a self-avoiding walk of length n, and the endpoints of the walk.

The ratios $\langle R^2\rangle_n / \langle R_e^2\rangle_n$ and $\langle R_m^2\rangle_n / \langle R_e^2\rangle_n$ are dimensionless, and each approaches a constant as $n \to \infty$:

$$\lim_{n\to\infty} \frac{\langle R^2\rangle_n}{\langle R_e^2\rangle_n} = \frac{\langle R^2\rangle}{\langle R_e^2\rangle}, \qquad \lim_{n\to\infty} \frac{\langle R_m^2\rangle_n}{\langle R_e^2\rangle_n} = \frac{\langle R_m^2\rangle}{\langle R_e^2\rangle}. \tag{21}$$

The limiting ratios are universal quantities (independent of the lattice). A conformal field theory prediction of Cardy and Saleur [13] relates these limiting ratios in two dimensions as follows:

$$\frac{246}{91}\frac{\langle R^2\rangle_n}{\langle R_e^2\rangle_n} - 2\frac{\langle R_m^2\rangle_n}{\langle R_e^2\rangle_n} + \frac{1}{2} = 0. \tag{22}$$

There are corrections to the scaling in the assumptions for c_n and R_n^2; these have been determined by field theoretic calculations which suggest that

$$\begin{aligned} c_n &= A\mu^n n^{\gamma-1}\left(1 + \frac{a_1}{n} + \cdots + \frac{b_1}{n^{\Delta_1}} + \frac{b_2}{n^{\Delta_1+1}} + \cdots\right), \\ R_n^2 &= C_R n^{2\nu}\left(1 + \frac{c_1}{n} + \cdots + \frac{d_1}{n^{\Delta_1}} + \frac{d_2}{n^{\Delta_1+1}} + \cdots\right), \end{aligned} \tag{23}$$

where corrections of the form a_i/n^i are said to be *analytic* and b_i/n^{Δ_1+i} are said to be *confluent* [29]. The confluent correction exponent Δ_1 above is the first in series of confluent correction exponents $\Delta_1 < \Delta_2 < \cdots$. Least squares fitting of R_n^2 or c_n to data to determine the exponents ν and γ may in some cases require regressions which include analytic and confluent corrections to scaling.

2.2.2. Fisher's scaling law for walks. The number of walks from the origin to a lattice vertex x is $c_n(x)$. For fixed values of x it is believed that

$$c_n(x) \sim B\mu^n n^{\alpha_s - 2}, \tag{24}$$

where α_s is the *entropic exponent* of lattice polygons. Normally, it is assumed that for fixed x and $n \to \infty$ the same asymptotic behaviour is seen for any x. Thus, by fixing x at the origin so that $c_n(0)$ is the number of polygons of length n rooted at the origin, one sees that $c_n(0) \sim C\mu^n n^{\alpha_s-2}$. For unrooted polygons, this shows that

$$p_n \sim B_p \mu^n n^{\alpha_s - 3}. \tag{25}$$

For this reason, α_s is often called the *polygon entropic exponent*.

The generating function of $c_n(x)$ is the *two-point function* defined by

$$G_t(0,x) = \sum_{n=0}^{\infty} c_n(x) t^n. \tag{26}$$

The radius of convergence of $G_t(0,x)$ is also given by $t_c = 1/\mu$. For values of $t < t_c$, the correlation length $\xi(t)$ is defined by

$$\xi^{-1}(t) = \liminf_{|x|\to\infty} \left[\frac{-\log G_t(0,x)}{|x|} \right], \tag{27}$$

where the inferior limit is taken over all sequences of points x such that $|x| \to \infty$.

The correlation function defines a length scale, which should be the same length scale set by other metric quantities. In other words, by determining the length scale set by the mean span in a model of walks where walks of length n are weighted by t^n, one gets

$$\langle R \rangle_t = \frac{\sum_n S_n c_n t^n}{\sum_n c_n t^n} \sim |\log(\mu t)|^{-\nu} \sim (1-\mu t)^{-\nu}. \tag{28}$$

Since the scaling of the correlation length will be the same as that of $\langle R \rangle_t$, we conclude that

$$\xi(t) \sim (1-\mu t)^{-\nu}. \tag{29}$$

$G_t(0,x)$ (as a function of x) will be significant when $|x|$ is comparable to the correlation length $\xi(t)$. Thus, there is a function g which decays asymptotically fast such that

$$G_t(0,x) \sim \frac{1}{|x|^{d-2-\eta}} g(|x|/\xi(t)). \tag{30}$$

This assumption introduces a new scaling exponent η, which is the *anomalous dimension*.

The susceptibility can be estimated from $G_t(0,x)$ by summing over x:

$$\chi(t) \simeq \sum_{x\in\mathbb{Z}^d} \frac{1}{|x|^{d-2-\eta}} g(|x|/\xi(t)) \sim \xi^{2-\eta} \sim (1-\mu t)^{-(2-\eta)\nu}. \tag{31}$$

Comparing this with equation (15) shows that

$$\gamma = (2-\eta)\nu. \tag{32}$$

This is Fisher's scaling relation.

2.2.3. Hyperscaling. Let c_{n_1,n_2} be the number of pairs of walk from the origin, of lengths n_1 and n_2, such that these walks intersect one another at least in one vertex different from the origin. It is believed that as both n_1 and n_2 go to infinity, then

$$c_{n_1,n_2} \sim A_{n_1,n_2} \mu^{n_1+n_2} n_1^{2\Delta_4+\gamma-2} F(n_1/n_2) \tag{33}$$

for some universal function F. This introduces the exponent Δ_4. This exponent should not be confused with the confluent correction exponent with the same symbol which was discussed following equations (23).

The exponent Δ_4 is thought to satisfy the hyperscaling relation

$$\mathrm{d}\nu - 2\Delta_4 + \gamma = 0 \tag{34}$$

in dimensions less than $d = 5$. To see this, argue as follows [100]: consider a walk of length n from the origin which should occupy a volume V of size $O(n^{\mathrm{d}\nu})$. A second walk of length n will intersect this walk if it is started from a vertex in V. This shows that

$$c_{n,n} \approx c_n^2 n^{\mathrm{d}\nu} \sim \mu^{2n} n^{2\gamma-2+\mathrm{d}\nu}. \tag{35}$$

A comparison of this with equation (33) gives $2\Delta_4 + \gamma - 2 = 2\gamma - 2 + \mathrm{d}\nu$ and this simplifies to the hyperscaling relation in equation (34).

Since the mean field value of $\Delta_4 = 3/2$, this relation fails in dimensions $d > 4$. One may show that in high dimensions ($d > 4$) that $\Delta_4 = 1 + \gamma/2$ [100]. In $d \geqslant 6$ dimensions it is known that $\Delta_4 = 3/2$ (see [100], this result is due to Hara and Slade).

The *interpenetration ratio* of a self-avoiding walk is defined by

$$\Psi_n = 2\left(\frac{d}{12\pi}\right)^{d/2}\left[\frac{c_{n,n}}{c_n^2\langle R^2\rangle_n^{d/2}}\right]. \tag{36}$$

Ψ_n measures the 'degree of hardness' of self-avoiding walks. The larger its value, the more difficult it is for two walks starting at the origin (say) to stay in roughly the same volume of space. By substituting the usual scaling relations for the quantities in Ψ_n, one finds that

$$\Psi_n \sim n^{2\Delta_4 - \gamma - \mathrm{d}\nu}. \tag{37}$$

In the case that hyperscaling as in equation (34) is satisfied, $\Psi_n \to$ constant as $n \to \infty$; violations of hyperscaling will either take Ψ_n to zero or to infinity. Computing this quantity is a sensitive test for hyperscaling [98].

The entropic exponent α_s is believed to be related to the metric exponent ν via a hyperscaling relation

$$2 - \alpha_s = \mathrm{d}\nu. \tag{38}$$

This relationship can only be proven by making significant assumptions, as pointed out in [100].

Assume first that $c_n(x)$ has the same scaling behaviour for any fixed lattice site x as $n \to \infty$. The number of pairs of walks (s_1, s_2), each of length n starting at the origin which avoid one another but which end in the same lattice site (say x) to form a closed self-avoiding ring (or polygon) is equal to the number of polygons of length $2n$ rooted at the origin:

$$c_{2n}(0) = \sum_{x\in\mathbb{Z}^d}\sum_{s_1,s_2} I(s_1 \cap s_2 = \{0, x\}). \tag{39}$$

In this expression, the summation is over all points x in the lattice, and over all walks s_1 and s_2 of length n each from the origin and ending in x. The indicator function $I(\cdot)$ is one if the intersection between s_1 and s_2 are their endpoints $\{0, x\}$, and zero otherwise.

Next assume that the main contribution to the summation above occurs when the distance $|x|$ is of order n^ν, then there are $O(n^{\mathrm{d}\nu})$ significant terms in the sum above.

The probability that two walks from the origin avoid one another is given approximately by c_{2n}/c_n^2, and the main contribution to this comes from steps close to the origin. The probability that the two walks from the origin and both ending in x both avoid one another is the square of this probability, $\left[c_{2n}/c_n^2\right]^2$. Thus, this probability scales as $[n^{1-\gamma}]^2$.

The final assumption is that for $|x|$ of order n^ν the probability that an n-step walk ends at x is inversely proportional to the volume of the sphere of radius n^ν: thus $c_n(x) \sim \mu^n n^{\gamma-1} n^{-\mathrm{d}\nu}$. Putting this together with the estimate in the previous paragraph shows that the summations over s_1 and s_2 above scale as $c_n(x)\left[c_{2n}/c_n^2\right]^2 \sim [\mu^n n^{\gamma-1} n^{-\mathrm{d}\nu}]^2[n^{1-\gamma}]^2$ when x is of $O(n^\nu)$.

Thus, since we assumed that the main contributions to the summation over x comes from $x = O(n^\nu)$, the number of significant terms is $O(n^{\mathrm{d}\nu})$, and one may estimate $c_{2n}(0)$ to be given by

$$c_{2n}(0) \sim n^{\mathrm{d}\nu}[\mu^n n^{\gamma-1} n^{-\mathrm{d}\nu}]^2[n^{1-\gamma}]^2 = \mu^{2n} n^{-\mathrm{d}\nu}. \tag{40}$$

On the other hand, it follows from equation (24) that $c_{2n}(0) \sim \mu^{2n}(2n)^{\alpha_s - 2}$. A comparison with the last equation gives Josephson's scaling law which relates the metric exponent to the polygon entropic exponent.

Table 1. Self-avoiding walk exponents.

d	2	3	Mean field
γ	43/32 [29]	1.1608(3) [43, 95–97]	1
ν	3/4 [32]	0.5877(6) [43, 98]	1/2
η	5/24 [32]	0.031(4) [14]	0
α_s	1/2 [32]	0.237 ± 0.002 [48, 97]	0
Δ_4	155/92	1.4603(12)	3/2
Δ_1	3/2 [15, 83]	0.47 ± 0.025 [48, 97]	

2.2.4. Numerical testing of scaling in the self-avoiding walk. The calculation of μ, critical exponents and testing of scaling and hyperscaling relations are major motivating factors in both the discovery of Monte Carlo methods for generating walks and the numerical simulation of walks.

In table 1, some exact values of critical exponents calculated by conformal field theory and Coulomb gas techniques are given in two dimensions [13, 29]. The estimates in three dimensions were computed by Monte Carlo simulations for γ and ν, while η was determined by the ϵ-expansion. Citations are given in square brackets.

In four and higher dimensions the exponents take on their mean field values, but in four dimensions the scaling laws are modified by logarithmic factors, for example, it is expected that

$$c_n \sim A\mu^n [\log n]^{1/4} \qquad \text{and} \qquad R_n^2 \sim C_R n [\log n]^{1/4} \tag{41}$$

in four dimensions, since the mean values of γ and ν are $\gamma = 1$ and $\nu = 1/2$.

The usual scaling assumptions for c_n (equation (13)) and R_n^2 (equation (20) are modified by analytic and confluent correction terms as in equation (23). These corrections must be taken into account when data analysis is done in order to extract the values of exponents, and every observable with a scaling law is subject to such corrections.

In two dimensions the exponent γ has been estimated using a variety of numerical methods. Exact enumeration of walks in the square lattice shows that

$$\gamma = 1.343\,745 \pm 0.000\,015 \tag{42}$$

in two dimensions; see [83], and for more details [54]. Estimating γ by Monte Carlo means is not nearly this accurate. Simulations in [130] give $\gamma = 1.345 \pm 0.004$.

The entropic exponents have also been estimated in three dimensions in addition to the field theoretic estimates in table 1. The estimate

$$\gamma = 1.1575 \pm 0.0006 \tag{43}$$

is given in [14]. Earlier estimates include $\gamma = 1.1608 \pm 0.0003$ by Monte Carlo simulation in [43] and $\gamma = 1.161 \pm 0.001$ in [53]. Using the PERM algorithm on the Domb–Joyce model gave the very accurate estimate $\gamma = 1.1573 \pm 0.0002$, obtained by Hsu *et al* [68].

The values for Δ_4 in table 1 were computed from the hyperscaling relation in equation (34). The combination $2\Delta_4 + \gamma$ was computed from intersecting self-avoiding walks in two and three dimensions in [98]. In two dimensions, the result is

$$2\Delta_4 + \gamma = 1.4999 \pm 0.000\,20. \tag{44}$$

The entropic exponent of polygons, α_s, has also been measured. Series analysis for polygons in two dimensions [84, 85] give α_s to a very high precision:

$$\alpha_s = 0.500\,0005 \pm 0.000\,0010. \tag{45}$$

Slight improvements on this result can be found in [82]. Monte Carlo estimates of α_s include the simulation of polygons with an atmospheric statistic. This gives [78]

$$\alpha_s = 0.532 \pm 0.027. \tag{46}$$

The exponent α_s has also been estimated from conformal field theory and Coulomb gas methods, which give the exact value $\alpha_s = 1/2$ in two dimensions [114, 115].

In three dimensions (see [19]) the estimate $\alpha_s \approx 0.24$ is given, while Monte Carlo methods using an atmospheric statistic gives 0.248 ± 0.016 in [78].

The metric exponent ν has been computed in numerous studies. The two-dimensional exact value $\nu = 3/4$ is due to conformal field theory calculations [32]. This was tested by exact enumeration studies [83] giving the estimate

$$2\nu = 1.500\,002 \pm 0.000\,003. \tag{47}$$

The hyperscaling relation $\mathrm{d}\nu = 2 - \alpha_s$ appears to be valid for $2 \leqslant d \leqslant 4$ for walks. In two dimensions, the exact values for ν and α_s in table 1 are consistent with hyperscaling in two dimensions. The value of ν has also been determined by Monte Carlo methods in two dimensions, for example, the estimate

$$\nu = 0.749\,63 \pm 0.000\,08 \tag{48}$$

is given in [98]. Comparison with the result in equation (44) gives accurate support for the hyperscaling relation in equation (34). In addition, comparing this result with equation (46) shows that these results are consistent with the hyperscaling law in equation (38).

In three dimensions many more Monte Carlo studies have been done to measure ν. The Flory value of $\nu = 3/5$ [39] is not exact, and the Edwards model [36] (to sixth order in the interaction parameter) gives the estimate $\nu \approx 0.588$ [22]. Series enumeration in three dimensions [53] gave the estimate

$$\nu = 0.592 \pm 0.003. \tag{49}$$

The pivot algorithm was used to estimate

$$\nu = 0.5909 \pm 0.0003 \quad \text{in [37]}, \tag{50}$$

$$\nu = 0.5877 \pm 0.0006 \quad \text{in [98]}. \tag{51}$$

The PERM algorithm on the Domb–Joyce model gave the improved estimate $\nu = 0.587\,65 \pm 0.000\,20$ [68]. Grassberger [43] reported the high-quality estimate

$$\nu = 0.585 \pm 0.0015 \tag{52}$$

by analysing data obtained by PERM. See [22, 37, 102, 129] for additional estimates of ν in three dimensions.

Estimates of the limiting ratios in equation (21) can be found in [98], where the relation in equation (22) is tested numerically. In two dimensions,

$$\frac{\langle R^2 \rangle}{\langle R_e^2 \rangle} = 0.140\,264 \pm 0.000\,073, \qquad \frac{\langle R_m^2 \rangle}{\langle R_e^2 \rangle} = 0.439\,605 \pm 0.000\,38, \tag{53}$$

and in three dimensions

$$\frac{\langle R^2 \rangle}{\langle R_e^2 \rangle} = 0.1599 \pm 0.0002. \tag{54}$$

The results in two dimensions can be substituted into equation (22) for a test of the conformal field theory prediction by Cardy and Saleur [13].

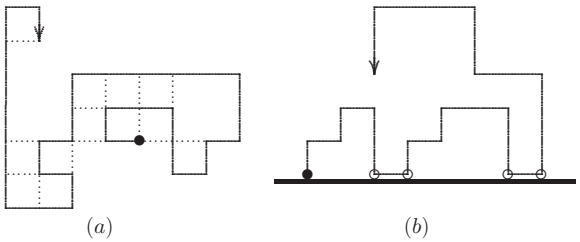

Figure 4. (*a*) A self-interacting self-avoiding walk. Nearest neighbour pairs of vertices joined by dotted lines are *contacts*. The number of contacts in this walk is 13. The *energy* of this walk is $c = 13$. (*b*) An adsorbing walk in two dimensions. The walk adsorbs on the horizontal line by interacting with it—each vertex of the walk in the adsorbing line is marked by a ○ and is called a *visit*. The *energy* of this walk is $v = 4$ (the origin is not a visit by convention).

The interpenetration ratio $\lim_{n\to\infty} \Psi_n = \Psi^*$ (see equation (36) was similarly estimated in [98], and Ψ^* was found to be finite in both two and three dimensions:

$$\Psi^* = \begin{cases} 0.662\,96 \pm 0.000\,43, & \text{in two dimensions,} \\ 0.2471 \pm 0.0003, & \text{in three dimensions.} \end{cases} \tag{55}$$

This is a sensitive test of the hyperscaling law in equation (34).

The confluent correction to scaling exponent Δ_1 in two dimensions was predicted to have values either $\Delta_1 = 3/2$ by Coulomb gas methods [114, 116] or $\Delta_1 = 11/16$ by conformal invariance methods [134]. Exact enumeration studies suggest strongly that $\Delta_1 = 3/2$ [83] and estimates using Monte Carlo methods [15] back this claim up. Some lower estimates can be found in the literature in [24]. In three dimensions only effective values of Δ_1 have been used in Monte Carlo simulations to estimate other exponents. It is often assumed that $\Delta_1 \approx 0.5$ in regressions (see for example [121]). Numerical estimates for Δ_1 can be found in [98]: $\Delta_1 = 0.56 \pm 0.03$, while Belohorec and Nickel estimated that $\Delta_1 = 0.515 \pm 0.007$ in an unpublished Guelph University Preprint. Further estimates for Δ_1 can be found in [23].

2.3. Interacting models of the self-avoiding walk

A self-avoiding walk model of a linear polymer typically introduces energy terms in the partition function of walks. For example, in a self-avoiding walk model of a self-interacting polymer the number of nearest-neighbour *contacts* will be the energy of the walk; see figure 4(*a*). This is a model of *collapsing walks*. Similarly, a self-avoiding walk model of a polymer adsorbing on a surface would be a walk in a half-space with energy equal to the number of vertices of the walk visiting the adsorbing surface (these are *visits*). Such a model is illustrated in figure 4(*b*), and it is a model of an *adsorbing walk*.

The microcanonical density of these models is the function $c_n(m)$, which is the number of self-avoiding walks of length n and energy m (where m is the number of nearest neighbour contacts or visits in a given walk), see for example, figure 4. The partition functions of these models are defined by

$$Z_n(z) = \sum_m c_n(m) z^m, \tag{56}$$

where $z = e^\beta$ is the *activity* or β is a *fugacity* conjugate to the energy.

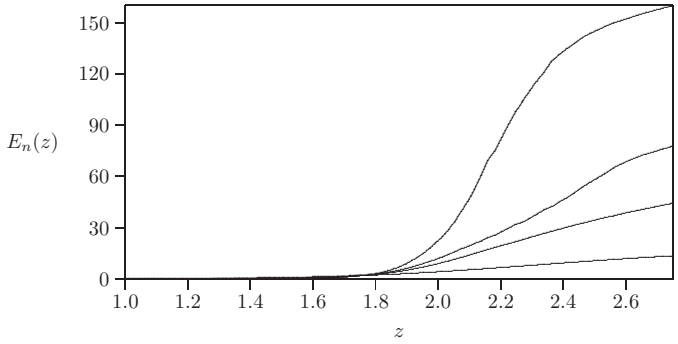

Figure 5. Energy (the mean number of visits) data for adsorbing walks in two dimensions, generated by the pivot algorithm in a multiple Markov chain implementation consisting of 10 Markov chains spaced evenly along the $\log z$-axis in $[0, 1]$. The curves are for walks of lengths 30 edges, 90 edges, 120 edges and 200 edges, from the bottom up.

In some models (for example in a model of adsorbing walks) it is possible to prove that the limiting free energy

$$\mathcal{F}(z) = \lim_{n\to\infty} \frac{1}{n} \log Z_n(z) \tag{57}$$

exists, is convex and therefore differentiable almost everywhere [100]. The existence of this limit is unknown for collapsing walks, and in those cases its existence is assumed as a working hypothesis, or the limit is replaced by a limit superior. Critical points in these models correspond to non-analytic points in the limiting free energy.

The generating function is defined by

$$G(t, z) = \sum_{n\geqslant 0} Z_n(z) t^n \tag{58}$$

and the radius of convergence of $G(t, z)$ is related to the limiting free energy by

$$t_c(z) = \lim_{n\to\infty} [Z_n(z)]^{-1/n} = e^{-\mathcal{F}(z)} \tag{59}$$

and in cases where this limit does not exist, it is replaced by the limit inferior.

2.3.1. Adsorbing walks. A positive self-avoiding walk in a half-space interacting with an infinite line or plane as illustrated in figure 4(*b*) is a model of an *adsorbing walk*. Each vertex of the walk in the adsorbing line or plane is a *visit*, and the partition function of the model is given by

$$Z_n(z) = \sum_{v} c_n^+(v) z^v, \tag{60}$$

where v is the number of visits of the positive walk to the adsorbing line or plane, and $c_n^+(v)$ is the number of positive walks from the origin of length n making v visits [26, 27, 66, 106, 108].

In this model, it is known that the free energy $\mathcal{F}(z) = \lim_{n\to\infty}[\log Z_n(z)]/n$ exists for all finite values of $z \geqslant 0$ [60]. The free energy of this model is singular at a critical point z_c corresponding to an adsorption transition. In figure 5, the mean energy (number of visits to the adsorbing plane) $E_n(z) = \frac{\mathrm{d}}{\mathrm{d}\log z} \log Z_n(z)$ of an adsorbing walk in the two-dimensional half-lattice with $Z \geqslant 0$ is plotted as a function of z. The number of visits remains low, but

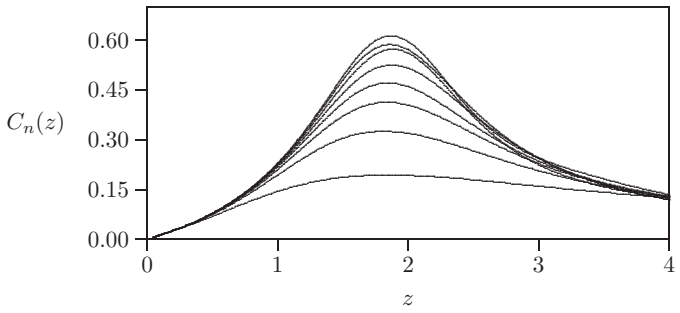

Figure 6. Collapsing two-dimensional walk specific heat curves $C_n(z)$ for $10 \leqslant n \leqslant 80$ in steps of 10 by a micro-canonical implementation of GARM (see section 10). The partition function of this model is given by $Z_n(z) = \sum_m c_n(m)z^m$, where $c_n(m)$ is the number of walks of length n with m nearest neighbour contacts. The algorithm produced an approximate enumeration for $c_n(m)$, from which the specific heat curves were computed.

increases quickly starting at $z \approx 1.9$. See section 2.6.1 for more details about the adsorbing transition in this model.

2.3.2. Collapsing walks. An interacting model of walks with nearest neighbour contacts weighted by z in the partition function is a model of *collapsing walks* [25]. In figure 4(*a*), a walk is illustrated with nearest neighbour contacts denoted in dotted lines. Each contact is assigned a weight z and the partition function of the model is

$$Z_n(z) = \sum_m c_n(m)z^m, \tag{61}$$

where $c_n(m)$ is the number of walks from the origin of length n steps and with m nearest neighbour contacts. It is not generally known that the limit $\mathcal{F}(z) = \lim_{n\to\infty}[\log Z_n(z)]/n$ exists in this model, but for $0 < z \leqslant 1$ the existence of a limiting free energy is known [144].

This is a model of a collapsing polymer in a poor solvent: for large values of z the model exhibits a phase of collapsed walks, and for small values of z a phase of expanded walks. The critical point in this model occurs at the *θ-point* $z = z_c$ where linear polymers undergo a 'collapse transition' from an expanded coil form to a compact ball [27, 142].

In figure 6, the *specific heat* given by $C_n(z) = \frac{d^2}{d\log^2 z}[\log Z_n(z)]/n$ is plotted as a function of z for collapsing walks. There is a clear maximum in the curve, coinciding with increasing fluctuations in the mean number of contacts when the walk goes through a θ-point from an expanded to a globule conformation.

Walks at the θ-point are often said to be θ-walks, and exact values for the critical exponents have been determined in two dimensions [31]; see section 2.6.2. At the θ-point the self-repulsive forces between vertices due to self-avoidance and nearest neighbour self-attractive forces between vertices cancel each other to first order. The walk is said to exhibit θ-statistics at this point, which (to first order) is random walk statistics in $d \geqslant 3$ (but with logarithmic corrections to scaling in $d = 3$).

2.4. Walks in confining geometries

A polymer molecule in solution in a colloid is confined to interstitial spaces between the large colloid particles and loses entropy. This entropy loss results in a repulsive force on the

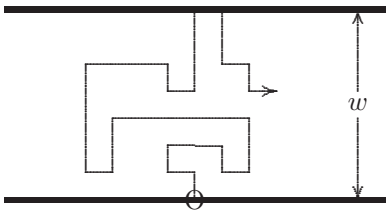

Figure 7. A self-avoiding walk in a slab or slit geometry.

confining particles. This is the basic mechanism of steric stabilization of colloid dispersions by a polymer adsorbed on the surfaces of the colloidal particles [21, 55, 111]. If the polymer is attracted to the two confining surfaces then the induced force can be attractive under some conditions. This is the basic mechanism of sensitized flocculation of colloids by adsorbed polymer chains [27].

This stabilization and flocculation of colloids can be modelled by a self-avoiding walk confined between two planes or plates as illustrated in figure 7.

In two dimensions, the model is a walk in a slit in the square lattice, and in three dimensions, it is a walk in a slab in the cubic lattice. Vertices in the walk interact with the bounding lines or plates in these models, and the most basic quantity is $c_n(v_1, v_2; w)$, the number of self-avoiding walks of length n from the origin making v_1 visits to the bottom bounding line or plane, and v_2 visits to the top bounding line or plane, of the slab $\mathbb{L}_w$ of width w defined by

$$\mathbb{L}_w = \{x \in \mathbb{Z}^d \,|\, 0 \leqslant Z(x) \leqslant w\}. \tag{62}$$

In this case $Z(x)$ is the Z-component or dth component of the vertex $x \in \mathbb{Z}^d$.

The partition function is given by

$$Z_n(z_1, z_2; w) = \sum_{v_1, v_2} c_n(v_1, v_2; w) z_1^{v_1} z_2^{v_2}. \tag{63}$$

Existence of the limiting free energy is known [73] for all $z_1 \geqslant 0$ and $z_2 \geqslant 0$:

$$\mathcal{F}_w(z_1, z_2) = \lim_{n\to\infty} \frac{1}{n} \log Z_n(z_1, z_2; L). \tag{64}$$

Moreover, $\mathcal{F}_w(z_1, z_2)$ is a strictly increasing function of w. The entropic force is defined by the discrete derivative of $\mathcal{F}$:

$$f_w(z_1, z_2) = \mathcal{F}_w(z_1, z_2) - \mathcal{F}_{w-1}(z_1, z_2) \tag{65}$$

and it is known that $f_w(z_1, z_2) > 0$ whenever $z_1 < 1$, or $z_2 < 1$, or both. In this case, there is a repulsive force between the bounding lines or plates of the slit or slab [73]. For large values of both z_1 and z_2 numerical simulations show that $f_w(z_1, z_2) < 0$ [72], and it is a conjecture that there is a force regime characterized by an attractive short ranged force between the boundaries of the slit of slab [73].

For the case $z_1 = z_2 = 1$ the scaling of $\mathcal{F}_w$ with w was examined by Daoud and de Gennes [21]. There are two competing length scales w and n^ν (where ν is the metric exponent) in the model. The extensive free energy should depend on a ratio of these length scales. That is

$$F_n(1, 1; w) \sim n^{-1} g(n^\nu / w), \tag{66}$$

where the function g is assumed to be a power law, say $g(x) = x^y$. Since $F_n(1, 1, w)$ becomes independent of n in the large n limit (and approaches the limiting free energy $\mathcal{F}_w(1, 1)$) this

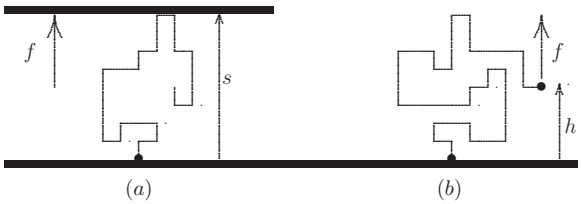

Figure 8. Two interacting models of stretched and pulled linear polymers. (*a*) A walk in a slab with vertices in both bounding plates of the slab is stretched if a force is applied to the top bounding plane of the slab so as to move these plates apart or together. In this model, the first vertex of the walk is fixed in the bottom bounding plane. This is a model of 'stretched walks'. (*b*) A walk in a half-space with first vertex fixed in the boundary of the space. A force applied to the last vertex of the walk may pull this vertex away from the boundary, or push it towards the boundary. We also call these 'pulled walks'.

implies that $y = 1/\nu$ and hence that $F_n(1, 1, w) \sim w^{-1/\nu}$. The (repulsive) force then scales as

$$f_w \sim w^{-1-1/\nu} \tag{67}$$

in this regime. This is expected to be the scaling form for all (z_1, z_2) less than the critical adsorption point z_c for adsorbing walks. Simulations in [72] provide strong numerical evidence for this scaling in the repulsive phase in this model.

2.4.1. Stretched walks. In figure 8(*a*) a model of a stretched walk is illustrated. The walk starts at the origin and is confined in a slab of width s with at least one vertex in the top bounding plate or line of the slab. If the Z-span of the walk s is defined by

$$S_z(s) = \max_{i,j}\{|Z(v_i) - Z(v_j)|\}, \tag{68}$$

where v_i are vertices in s with Z-component $Z(v_i)$, then the model is constrained to have Z-span equal to s. The partition function of stretched walks is given by

$$Z_n(f) = \sum_s c_n(s)e^{fs}, \tag{69}$$

where $c_n(s)$ is the number of walks in the slab with Z-span equal to s, and this defines the model of *stretched walks* [27]. The parameter f is an external force applied to the bounding plates of the slab and may be either positive (stretching) or negative (compressing). The free energy of this model is given by $\mathcal{F}(f) = \lim_{n\to\infty}[\log Z_n(f)]/n$, and this exists [76]. There is no phase transition in this model, but for large values of f the walk is stretched into a series of blobs (the Pincus regime [124]); this has been confirmed rigorously for values of f which are large enough [75, 76]; see [69] as well.

In figure 9, the mean Z-span (defined as the mean height or Z-span of the slab in figure 8) is plotted as a function of f. For negative values of f the bounding plates are squeezed together, and the mean Z-span is small. Increasing positive (or stretching) values of the force increases this, starting when $f \approx 0$.

2.4.2. Pulled walks. A model of pulled walks is illustrated in figure 8(*b*). A positive walk is fixed at one endpoint in the origin, while the other endpoint is subjected to an external force

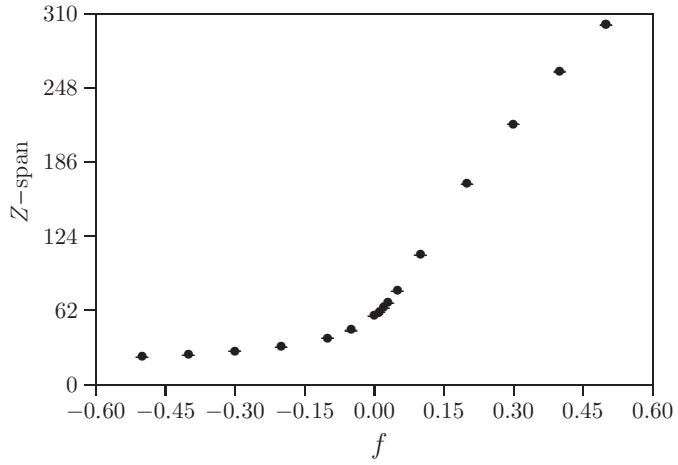

Figure 9. The Z-span of stretched walks of length $n = 1000$ plotted as a function of the applied force f. Negative forces correspond to pushing forces on the bounding plates in figure 8(*a*), while positive forces correspond to stretching forces on the top plate in figure 8(*a*). These data are collected by simulating stretched walks using a multiple Markov chain implementation of the pivot algorithm with elementary moves consisting of both one-point and two-point pivots.

in the vertical direction. If h is the height of the last vertex of the walk above the plane $Z = 0$, then the partition function of this model is given by

$$Z_n(f) = \sum_s c_n^+(h) e^{fh}, \tag{70}$$

where $c_n^+(h)$ is the number of positive walks from the origin with last endpoint at height h. This defines a model of *pulled walks*. The parameter f is an external force applied to the last vertex in the walk and may be either positive (pulling) or negative (pushing). The free energy of this model is given by $\mathcal{F}(f) = \lim_{n\to\infty}[\log Z_n(f)]/n$, and this exists [76]. Similar to stretched walks, it is known that pulled walks have a Pincus regime for values of $f \geqslant 0$ sufficiently large [69].

In figure 10, the mean height of the last vertex of a pulled walk, defined by $\frac{\mathrm{d}}{\mathrm{d}f} \log Z_n(f)$, is plotted as a function of f. For negative values of f this endpoint stays close to the bounding plane, but it increases quickly with f for pulling (positive) values of the applied force, starting at $f \approx 0$.

2.5. *Scaling in interacting models of walks*

Phase changes in interacting models of walk correspond to singularities in the limiting free energy $\mathcal{F}(z)$. The basic assumption is that the singular part of the free energy has a singularity described by

$$\mathcal{F}_s(z) \sim |z - z_c|^{2-\alpha}, \tag{71}$$

where α is the *specific heat* exponent (and should not be confused with the polygon entropic exponent α_s).

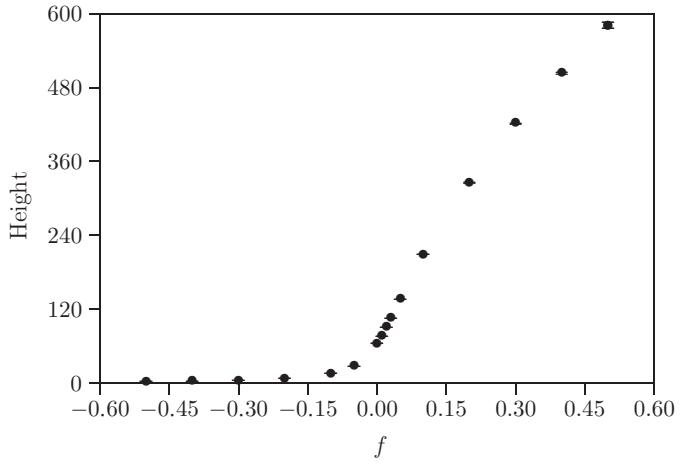

Figure 10. The height of the last vertex in a model of pulled walks (see figure 8(*b*)) of length $n = 2000$ plotted against f. Negative values of f correspond to forces pushing the endpoint towards the bounding plate at the bottom in figure 8(*b*), while positive forces correspond to forces pulling the endpoint of the walk away from the bottom plate in figure 8(*a*). These data are collected by simulating pulled walks using a multiple Markov chain implementation of the pivot algorithm with elementary moves consisting of both one-point and two-point pivots.

The value of the exponent α may be dependent on whether $z \to z_c^-$ or $z \to z_c^+$, in which case the model is said to be *asymmetric*. Normally, α describes the divergence in the curvature of $\mathcal{F}(z)$ at the critical point $z = z_c$: this is the specific heat defined by

$$\mathcal{C}(z) = \frac{\mathrm{d}^2\mathcal{F}}{\mathrm{d}z^2} \sim |z - z_c|^{-\alpha}. \tag{72}$$

The (metric) correlation length in the model is also a function of $|z_c - z|$, and it typically diverges as $z \to z_c$ as a power law

$$\xi(z) \sim (z_c - z)^{-\nu_\theta}, \tag{73}$$

where ν_θ is the *metric exponent* of the model on approach to the critical point. Since the free energy $\mathcal{F}(z)$ is a density, its singular part should scale ξ as $\mathcal{F}_s(z) \sim \xi^d$ in d dimensions. Substituting ξ into this and comparing it to equation (71) give the hyperscaling relation

$$2 - \alpha = \mathrm{d}\nu_\theta, \tag{74}$$

which is Josephson's scaling law. Testing the validity of this relation verifies the scaling assumptions in equation (71).

The critical exponents take their mean-field values above the upper critical dimension d_c of the critical point, and logarithmic corrections are normally expected when $d = d_c$; see for example the introduction of [45] for an in-depth discussion of this point. Hyperscaling (see equation (74)) is violated for $d > d_c$ when the exponents assume their mean field values.

2.5.1. Finite size scaling. *Finite size scaling* is the scaling of the partition function $Z_n(z)$ as $n \to \infty$. Define the *scaling field* $s = z_c - z$, then the free energy per vertex $F_n(z) = \frac{1}{n}\log Z_n(z)$ is a function of the compound variable $n^{\phi_c}s$

$$F_n(z) \simeq \frac{1}{n}\mu_d(n^{\phi_c}s), \tag{75}$$

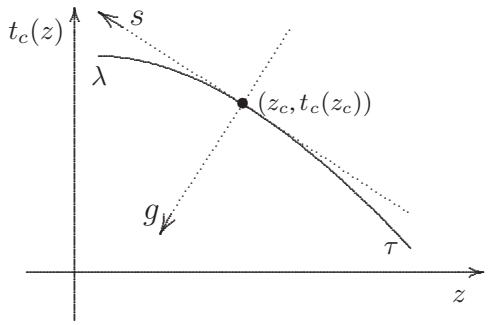

Figure 11. The tricritical point is located at the point marked •. At this point, the nature of the singularity in $G(t, z)$ changes from a simpler singularity along the critical curve $t_c(z)$ to a more complicated singularity. The scaling of the generating function around this point is given in terms of the ansatz in equations (80) and (81).

where $\mu_d(x)$ is a universal scaling function and ϕ_c is the *finite size crossover exponent*. Since $F_n(z) \to \mathcal{F}(z) \sim s^{2-\alpha}$ as $n \to \infty$ and $s \to 0$ by equation (71), it follows that $\mu_d(x) \sim x^{1/\phi_c}$ as $x \to 0$, and this shows that

$$2 - \alpha = 1/\phi_c. \tag{76}$$

By Josephson's law, this establishes the hyperscaling relation

$$2 - \alpha = d\nu_\theta = 1/\phi_c \tag{77}$$

between the finite size crossover exponent and the metric exponent. In analogy with equation (38) the metric exponent ν_θ is related to the scaling of clusters in the model as the critical point z_c is approached.

Critical scaling of the partition function is a generalization of the scaling ansatz in equation (13). In particular, one expects

$$Z_n(z) \sim A\mu_z^n n^{\gamma-1}, \tag{78}$$

where μ_z is now dependent on z, while the exponent γ should not change its value until the model passes through a phase change at some critical value $z = z_c$. At $z = z_c$ the value of the exponent γ changes, and in some models it takes the 'special value' γ^s at the 'special point' (or critical point). That is, it is generally assumed that

$$Z_n(z_c) \sim A\mu_{z_c}^n n^{\gamma^s-1} \tag{79}$$

and the value of γ^s has been estimated in a number of different models, in particular for adsorbing and collapsing walks, see for example [45, 66]. These assumptions are particularly valid below the upper critical dimensions d_c of the critical point, but become modified by logarithmic terms if $d = d_c$.

2.5.2. Homogeneity of the generating function. Critical behaviour in interacting models is best described by a tricritical ansatz [94]. The simplest implementation of a tricritical scaling ansatz is based on the homogeneity of the generating function: define the scaling fields $g = t_c(z) - t$ and $s = z_c - z$, where t_c is the critical curve defined in equation (59) and schematically illustrated in figure 11.

The fields define a coordinate system (s, g) in which the singularities in $G(t, z)$ lie along a critical curve which passes through the origin in the sg-plane, as illustrated in figure 11.

The nature of the singularity in $G(t, z)$ changes at the tricritical point. For $z < z_c$ (or $s > 0$) the generating function usually exhibits a simpler type of singularity along the critical curve $g_c(s)$, such as a pole or a branch point, while for $z > z_c$ in the collapse or adsorbed phase, the simpler singularity turns into a more complicated singularity (such as an essential singularity). The curve of simpler singularities composes the λ-curve, while the curve of more complicated singularities composes called the τ-curve.

Approaching the tricritical point in the gs-plane gives singular behaviour which depends on the direction of approach. For example, along the s-axis the singularity is assumed to be determined by the critical exponent $2 - \alpha_u$, so that the singular part of the generating function has behaviour given by

$$G_s(t, z) \simeq S_\lambda s^{2-\alpha_u}, \quad \text{along the } s\text{-direction.} \tag{80}$$

Consistent with this, the generating function is assumed to have singular behaviour described by the critical exponent $2 - \alpha_t$ if the tricritical point is approached along the g-axis. Thus[1]

$$G_s(t, z) \simeq S_\lambda g^{2-\alpha_t}, \quad \text{along the } g\text{-direction.} \tag{81}$$

Along the λ-line, away from the tricritical point, the singularity in G_s is described by the exponent $2 - \alpha_+$:

$$G_s(t, z) \simeq S_\lambda (t_c(z) - t)^{2-\alpha_+} \quad \text{on approaching the } \lambda\text{-line.} \tag{82}$$

Under scaling of the axes g and t by ℓ, G_s rescales as follows:

$$G_s(t, z) \simeq \ell^{-d} G_s(\ell^{y_g} g, \ell^{y_t} s) \tag{83}$$

in d dimensions, where y_g and y_t are *tricritical scaling exponents*. If ℓ is eliminated in this relation by putting $\ell = c \cdot g^{-1/y_g}$ for some constant c, then

$$G_s(t, z) \simeq g^{d/y_g} G_s(c, g^{-y_t/y_g} s). \tag{84}$$

Comparison with equation (81) indicates that along a curve $g^{-y_t/y_g} s = \text{const}$, as $g \to 0$, the singular part of G_s must be given by $g^{2-\alpha_t}$. Thus, it follows that

$$2 - \alpha_t = d/y_g. \tag{85}$$

The ratio y_t/y_g in the above describes the crossover scaling between the scaling fields g and s, and we define the *tricritical* crossover exponent $\phi = y_t/y_g$. This shows that

$$G_s(t, z) \simeq g^{2-\alpha_t} f_s(g^{-\phi} s), \tag{86}$$

where f_s is a scaling function. By comparison to equation (80) the exponent $2 - \alpha_u$ indicates scaling with respect to s; so we see that

$$G_s(t, z) \simeq s^{2-\alpha_u} f'_s(g^{-\phi} s), \tag{87}$$

where $f'_s(x) = x^{-(2-\alpha_t)/\phi)} f_s(x)$ and

$$\phi = \frac{2 - \alpha_t}{2 - \alpha_u}, \tag{88}$$

on approach of the tricritical point at the origin of the gs-plane.

The shape of the λ-line in the gs-plane close to the tricritical point is described by the exponent ψ (which we call the *shift exponent*):

$$t_c(z) \simeq t_c(z_c) + a_\lambda (z_c - z) + b_\lambda (z_c - z)^\psi \simeq b_\lambda s^\psi. \tag{89}$$

[1] The exponents $-\gamma_u = 2 - \alpha_u$ and $-\gamma_t = 2 - \alpha_t$ are frequently used in equations (80) and (81) instead. This assumes that $G_s(t, z)$ diverges at the tricritical point, which may not be the case in general.

Along the λ-line the argument of the scaling function cannot grow unbounded; this implies that $g^{-\phi}s \approx$ const along this line, showing that the shape of the λ-line in the gs-plane is approximated by $g_\lambda(s) \approx \text{const}\, s^{1/\phi}$. Comparison with equation (37) shows that

$$\psi = 1/\phi. \tag{90}$$

The (metric) length scales close to the tricritical point are determined by the correlation length ξ which scales on approach of the tricritical point along the g-axis as

$$\xi(g) \sim g^{-\nu_t}, \tag{91}$$

where ν_t is a *metric exponent*. Rescaling of the g-axis by ℓ will similarly rescale $\xi(g)$ by ℓ. Since ξ and ℓ are both metric quantities, ξ is proportional to ℓ and by eliminating g from equation (91), one sees that $\xi \sim \ell^{y_g \nu_t}$. Thus,

$$y_g = 1/\nu_t \tag{92}$$

and therefore from equation (85) we obtain

$$2 - \alpha_t = \mathrm{d}\nu_t. \tag{93}$$

Observe that $\nu_t \neq \nu$ and $\alpha_t \neq \alpha$ so that this relationship is not the same as Josephson's law given above. Instead, ν_t describes the metric scaling of clusters at the tricritical point in the model and relates this metric scaling to the thermodynamic tricritical exponent α_t.

A similar argument introduces a second tricritical metric exponent $1/y_t$ such that $2 - \alpha_u = d/y_t$.

In both finite size scaling and homogeneity of the generating function, we uncovered crossover behaviour between the scaling directions described by equations (75) and (86). If the scaling fields in finite size scaling are identified as s and $1/n$, and s eliminated from equation (86), then g and n can only be combined into a single variable which stays finite and non-zero as $g \to 0^+$ or $n \to \infty$ if $\phi = \phi_c$ (where ϕ_c is the finite size crossover exponent). This in particular gives the hyperscaling relation in equation (77) so that

$$2 - \alpha = \mathrm{d}\nu_\theta = 1/\phi. \tag{94}$$

This relates the thermodynamic tricritical crossover scaling exponent ϕ with the metric exponent ν_θ of the model at the tricritical point and the specific heat exponent α.

2.6. Numerical testing of tricritical scaling

2.6.1. Adsorbing walks. Tricritical scaling has been verified numerically for adsorbing walks in numerous studies. The crossover exponent for linear polymer adsorption is $\phi = 1/2$ in all dimensions. For example, numerical studies in [77] gives the estimates

$$\phi = \begin{cases} 0.501 \pm 0.015, & \text{if} \quad d = 2, \\ 0.5005 \pm 0.0036, & \text{if} \quad d = 3. \end{cases} \tag{95}$$

The mean field value is also $\phi = 1/2$ while conformal field theory predicts that $\phi = 1/2$ in the two-dimensional polymer adsorption problem [12]. Other numerical estimates in the literature scatter about $1/2$, for example, $\phi = 0.530 \pm 0.007$ [108] and $\phi = 496 \pm 0.004$ [66], $\phi = 0.50 \pm 0.01$ [45], amongst many others (but older simulations gave estimates rather larger than $1/2$; see for example [106]).

The exponent γ assumes its self-avoiding walk value in the desorbed phase and is also distinguished by the geometry of the walk: γ_1 is the value of the entropic exponent γ in equation (78) if only one endpoint is attached to the absorbing surface, while γ_{11} is the value

when both endpoints are constrained to be in the adsorbing surface, forming a loop. A remarkable relation involving γ_1 and γ_{11} is Barber's scaling relation [6], given by

$$2\gamma_1 - \gamma_{11} = \nu + \gamma. \tag{96}$$

In two dimensions, the exact values in table 1 give the value $2\gamma_1 - \gamma_{11} = 67/32 = 2.093\,75$. A numerical test was done in [7]: $\gamma_1 = 0.945 \pm 0.005$, $\gamma_{11} = 0.19 \pm 0.03$, so that $2\gamma_1 - \gamma_{11} = 2.08 \pm 0.04$.

In three dimensions numerical simulations in this phase in [66] give the estimates $\gamma_1 = 0.679 \pm 0.002$ and $\gamma_{11} = -0.383 \pm 0.005$, which combines with γ and ν to give $2\gamma_1 - \gamma_{11} - (\gamma + \nu) = -0.005 \pm 0.006$ (simulations in [43] give $\gamma = 1.1608 \pm 0.0003$ and $\nu = 0.585 \pm 0.0015$). The values of γ and ν in table 1 show that $\gamma + \nu = 1.7485 \pm 0.0009$, consistent with these results.

The value of the adsorption critical point z_c has also been estimated [77] in the square and cubic lattices:

$$z_c = \begin{cases} 1.759 \pm 0.018, & \text{if} \quad d = 2, \\ 1.334 \pm 0.027, & \text{if} \quad d = 3. \end{cases} \tag{97}$$

Other estimates for z_c in the two dimensions are somewhat inconsistent: for example, in [106] it is found that $z_c = 2.059 \pm 0.012$. In three dimensions, the value of z_c has been consistently shown to be close to the estimate above. For example, $z_c = 1.338 \pm 0.005$ [108], $z_c = 1.3310 \pm 0.0003$ [66] and $z_c = 1.338 \pm 0.065$ [152].

Assuming that hyperscaling $2 - \alpha = \mathrm{d}\nu_\theta = 1/\phi$ holds in these models, one obtains $\alpha = 0$ in both two and three dimensions. Estimates of the exponent $2 - \alpha_t$ in equation (81) have been determined in [40]: in this case $2 - \alpha_t = -1.46 \pm 0.004$.

A slightly different implementation of this model is to count adsorbing edges (rather than visits) in the adsorbing plane [45]. This model is in the same universality class, but the location of the critical point is shifted away from the estimates above. In two dimensions the best estimate of the critical point is $z_c = 2.036 \pm 0.002$ [45].

2.6.2. Collapsing walks. The θ-point and collapsing walks have received much attention using Monte Carlo simulations in the literature [17, 45, 144, 145]; the situation is slightly more complicated than for collapsing walks and this model is numerically more challenging.

The upper critical dimension for the θ-transition in walks is $d_c = 3$. Thus, for $d \geqslant 3$ the model has mean-field critical exponents (for example, the crossover exponent takes value $\phi = 0.5$), with logarithmic corrections to scaling laws. Numerical data in three dimensions are consistent with $\phi = 0.5$ in three dimensions [144, 145].

In two dimensions there are two models for θ polymers. The first model is the θ model described in section 2.3.2, while the second model introduced in [34] is referred to as the θ' model (see also [20]). The θ' model is a model of polymers in a hexagonal lattice with growing percolation clusters driving a collapse transition at the percolation threshold—these are θ' polymers and numerous studies have been done to determine if they belong to the same universality class as θ polymers [17, 35, 125, 138].

The crossover exponent ϕ for two-dimensional walks at the θ-point has been determined by exact enumeration in [103]: $\phi = 0.90 \pm 0.02$, while other studies gave values which are significant smaller, for example $\phi = 0.64 \pm 0.05$ [128] (exact enumeration), $\phi = 0.53 \pm 0.004$ [17] (Monte Carlo), $\phi = 0.48 \pm 0.07$ [133] (transfer matrix), $\phi = 0.66 \pm 0.02$ [107] (Monte Carlo) and $\phi = 0.52 \pm 0.07$ [136, 137] (Monte Carlo), amongst many other results. Generally, this model is numerically difficult.

The locations of the θ-point in two dimensions [40] and three dimensions [144, 145] have been computed to be

$$z_c = \begin{cases} 1.93 \pm 0.03, & \text{if} \quad d = 2, \\ 1.3204 \pm 0.0055, & \text{if} \quad d = 3. \end{cases} \tag{98}$$

These values are consistent with other studies, for example $z_c \approx 1.93$ [17], $z_c = 1.931 \pm 0.012$ [107], $z_c = 1.95 \pm 0.11$ [103] and $z_c \approx 1.90$ [10], all in the square lattice.

The values of the exponent $2 - \alpha_t$ in equation (81) for collapsing walks have been determined in [40]: in this case $2 - \alpha_t = -0.57 \pm 0.02$. The metric exponent was estimated in [103]: $\nu \approx 4/7$ in two dimensions and $\nu = 0.57 \pm 0.015$ while $\gamma = 1.075 \pm 0.04$ [138].

The upper critical dimension for collapsing walks is $d = 3$, and critical exponents in $d = 3$ takes their mean field values, while scaling laws are modified by logarithmic terms. Simulations in [46] suggest that these logarithmic corrections are additive, rather than multiplicative at the θ-point.

Conformal field theory have also been used to determine exponents in the θ'-model [34]. In two dimensions the exact values have been determined in [34]: $\phi = 3/7$, $\gamma = 8/7$, $\nu = 4/7$ and [150] $\gamma_1 = 4/7$.

The *special θ-point* (not to be confused with the θ'-point) is the (higher order) multicritical point for collapsing polymers which are critical with respect to adsorption [150]. The model is usually defined in terms of 'surface' and 'bulk' activities, but these can be readily put into visits and contacts in the microcanonical ensemble.

The special values of γ_1 and γ_{11} when $z = z_c$ (a the θ-point) have also been estimated in [45]; in two dimensions $\gamma_1^s = 1.46 \pm 0.01$ and $\gamma_1^s - \gamma_{11}^s = 0.64 \pm 0.01$. This gives $2\gamma_1^s - \gamma_{11}^s = 2.10 \pm 0.014$ [45], in agreement with Barber's relation since $67/32 = 2.094$. The exact value of γ_1^s is $\gamma_1^s = 93/64$ [49]. At the special point in $d = 3$ dimensions, numerical simulations by Hegger and Grassberger [66] gave the values $\gamma_1^s = 1.230 \pm 0.002$ and $\gamma_{11}^s = 0.714 \pm 0.006$ so that $2\gamma_1^s - \gamma_{11}^s = 1.746 \pm 0.008$. The estimates in table 1 show that $\gamma + \nu = 1.7485 \pm 0.0009$, consistent again with Barber's scaling relation, this time at the special point.

In contrast to the special θ-point, the *special θ'-point* is a multicritical point in the θ' model, but critical with respect to adsorption. Exact values in two dimensions have been determined for this model by Vanderzande *et al* [138, 150, 151] giving $\phi_s = 8/21$, $\gamma_1 = 4/7$, while their exact enumeration study shows that $\gamma_1 = 0.57 \pm 0.02$ (see also [151]). Exact enumeration in [40] gives $\phi_s = 0.40 \pm 0.05$.

It is also proposed in [122] that a surface tension term will further modify scaling of the partition function in models where a phase is characterized by dense walks. A droplet analysis suggests

$$Z_n(z) \sim A(\mu_z)^n \mu_1^{-n^\sigma} n^{\gamma-1}, \tag{99}$$

where $\sigma = (d-1)/d$ and $\mu_1 > 1$, for $z > z_c$ in the dense phase. This scaling form is also obtained for two-dimensional dense walks (with $d = 2$, and $\sigma = 1/2$) [6, 34], and the exponent $\gamma = 0.92 \pm 0.09$ was computed in [123] for a model of collapsing walks with a dense phase. Numerical verification of the presence of surface tension terms in the collapse phase in two dimensions can be found in [67].

2.6.3. Stretched and pulled walks. Simulations of *stretched and pulled walks* [47, 76] demonstrate the appearance of a Pincus phase for positive (stretching or pulling) values of the applied force f. Pulled walks have been proven to be ballistic for $f > 0$ with a non-zero density of cut-planes which partition the walk into Pincus-type balls in [69], while a slightly

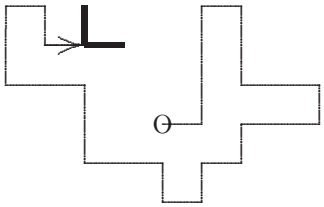

Figure 12. The positive endpoint atmospheric edges of this walk in the square lattice are indicated by the bold edges. In this example, $a^e_+ = 2$. The last edge in this walk (denoted by the arrow) is its negative atmospheric edge. In this case, the neutral endpoint atmospheric statistic a^e_0 is the number of ways in which that last edge can be removed and put back to obtain distinct walks. In this example, $a^e_0 = 1$.

weaker result for stretched polygons can be found in [75]. More numerical results can be found in [76]. The simulations in [47] are of walks in a poor solvent and a transition from collapsed globules to stretched conformations occur at a critical value of the applied force.

3. Self-avoiding walk atmospheres

In this section, we define the *atmospheres* of self-avoiding walks. These are best seen as potential constructions that can be executed on a walk to increase or decrease its length, or to change its conformation. There are a wide variety of possible atmospheres: the simplest would be to consider the ways in which a walk can be extended by one edge by adding an edge to its endpoint, or by inserting it somewhere along the walk. More complicated atmospheres are defined by reflecting or rotating parts of a given walk to change its conformation. The number of different atmospheres of a given class of a given walk is an *atmospheric statistic*, and it may be closely associated with c_n, as we shall see below. Atmospheres were defined in [78, 130, 131].

3.1. Positive and negative atmospheres

A *positive atmosphere* of a self-avoiding walk is composed of all the ways in which a walk can be extended by adding a fixed number of edges in a defined way. Similarly, we define a *negative atmosphere* by the ways in which a walk can be shortened by removing a fixed number of edges from it. Below, we define some particular atmospheres.

(1) *Endpoint atmospheric edges.* Suppose that s is a self-avoiding walk from the origin of length n and last vertex v_n. A *positive endpoint atmospheric edge* of s is a lattice edge incident with v_n which can be appended to s to create a new walk of length $n + 1$. The *size* $a^e_+(s)$ of the positive endpoint atmosphere of s is the number of positive atmospheric edges. The last edge in s is its *negative endpoint atmospheric edge* since its removal will reduce the length of the walk by one edge. The *size* of the negative atmosphere is defined to be $a^e_-(s) = 1$ trivially, unless s is the trivial walk $\emptyset$ of length zero, with $a^e_-(\emptyset) = 0$. The endpoint atmosphere of a walk is illustrated in figure 12. Observe that every positive atmospheric edge becomes a negative atmospheric edge when added to the walk. Thus, positive and negative atmospheres are reversible into one another.

A *neutral endpoint atmospheric statistic* can also be defined for a walk: this is the number of distinct ways the last edge of a walk can be removed and then reattached to the walk to give a new walk. For example, in the example in figure 12 the last edge can be removed

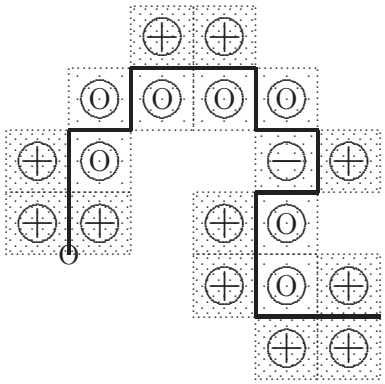

Figure 13. The plaquette atmospheres of a self-avoiding walk. Positive atmospheric plaquettes are denoted by a +, negative atmospheric plaquettes by − and neutral atmospheric plaquettes by *O*. In this example, $a_+^p = 11$, $a_o^p = 7$ and $a_-^p = 1$.

and put back in only one way, so that $a_0^e(s) = 1$ is the neutral atmospheric statistic of this particular walk.

(2) *Atmospheric plaquettes.* See figure 13: if s is a self-avoiding walk of length n from the origin, then three successive edges in a ⊔-conformation is a *negative atmospheric plaquette*. Conversely, if an edge in s can be replaced by three edges in a ⊔-conformation to create a new self-avoiding walk of length $n+2$, then the edges form a *positive atmospheric plaquette*. Two adjacent edges incident at 90° with one another and bounding a unit square with exactly two edges and three vertices in the walk is a *neutral atmospheric plaquette*.

The *size* of the negative atmosphere of s is denoted by $a_-^p(s)$ and is the number of occurrences of negative atmospheric plaquettes along s. The *size* of the positive atmosphere of s is denoted by $a_+^p(s)$ and is the number of occurrences of positive atmospheric plaquettes along s. The size of the neutral plaquette atmosphere $a_o^p(s)$ is similarly defined.

(3) *Generalized atmospheric edges.* Suppose that s is a self-avoiding walk of length n. An edge in s is *contracted* by deleting it, and then concatenating the two subwalks incident on its two endpoints. This may, or may not, result in a new walk.

Suppose a is an edge in s and s' is the object obtained by *contracting* a. Then a is a *negative generalized atmospheric edge* if s' is a self-avoiding walk of length $n-1$. Negative generalized atmospheric edges are illustrated in figure 14. On the other hand, if e is a lattice edge and s is cut in two subwalks s_1 and s_2 at a vertex v_j, then e is a *positive generalized atmospheric edge* if the object $s_1 e s_2$, constructed by concatenating e onto the endpoint of s_1 and translating s_2 so that its first vertex is concatenated on the other endpoint of e, is itself a self-avoiding walk of length $n+1$. The *size* of the negative atmosphere of s is denoted by $a_-^g(s)$ and it is the number of edges which are negative atmospheric edges. The *size* of the positive atmosphere of s is the number of positive atmospheric edges and is denoted by $a_+^g(s)$. Examples of positive generalized atmospheric edges are illustrated in figure 15.

Other definitions for atmospheres can be added to the list, but in this review the focus will be on the above. Positive atmospheric edges on a walk s of length n creates a linkage between s and walks s' of longer length. We denote the linkage by (s, s'), and observe that s'

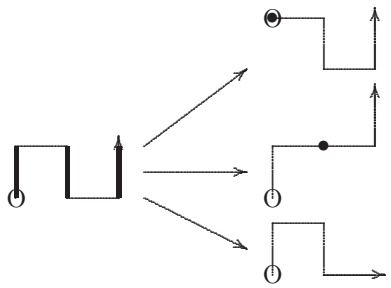

Figure 14. The negative generalized atmosphere of a walk. The bold edges on the left are negative generalized atmospheric edges. By contracting these edges the walks on the right-hand side are obtained. The location of the contracted edge in each case is marked with a •. This walk has size of its negative generalized atmosphere $a^g_- = 3$.

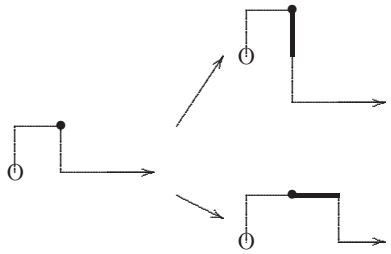

Figure 15. The positive generalized atmosphere of a walk. The vertex marked with a • in the walk on the left-hand side has two generalized atmospheric edges incident with it; one point in the south direction and the other pointing in the east direction. By inserting an edge in the south direction at this vertex, the top walk is obtained. If an edge in the east direction is inserted at this vertex, then the walk at the bottom is obtained. The inserted positive atmospheric edges are indicated in bold. This walk has size of its positive generalized atmosphere $a^g_+ = 14$.

can be obtained from s by adding positive atmospheric edges to s (and in the case of general atmospheres, concatenate subwalks suitably) to obtain s'.

Observe that in any linkage (s, s'), the walk s can be obtained from s' by removing negative atmospheric edges to reverse positive atmospheric moves which took s to s'.

The two walks in each linkage (s, s') are, in fact, connected uniquely by specified positive atmospheric edges in s which gives rise to specific negative atmospheric edges in s'.

The linkages set up correspondences between walks of length n and $n + 1$ (for endpoint and general atmospheres) and n and $n + 2$ (for plaquette atmospheres). This is illustrated in schematically in figure 16. The number of linkages in this figures can be obtained either by counting positive atmospheres of walks of length n, or negative atmospheres from walks of length $n + 1$ (or $n + 2$ for plaquette atmospheres).

This shows that for endpoint atmospheres,

$$\text{Number of linkages} = \sum_{|s|=n} a^e_+(s) = \sum_{|s|=n+1} a^e_-(s) = c_{n+1} \tag{100}$$

since $a^e_-(s) = 1$ if s has length at least equal to 1. If the average size of positive atmospheres of walks of length n is denoted by $\langle a^e_+ \rangle_n$, then it follows from the above that

$$\langle a^e_+ \rangle_n = \frac{c_{n+1}}{c_n}. \tag{101}$$

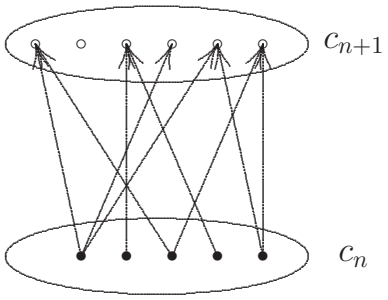

Figure 16. Atmospheric edges or plaquettes set up linkages between walks of length n and $n+1$ (or $n+2$). These linkages are denoted schematically by arrows above. An arrow from a walk of length n (denoted by a $\bullet$) to a walk of length $n+1$ (denoted by a $\circ$) corresponds to a positive atmospheric move, which can be reversed by a negative atmospheric move. By counting the number of linkages, one obtains equations (101)–(103).

A similar argument shows that

$$\frac{\langle a_+^p \rangle_n}{\langle a_-^p \rangle_{n+2}} = \frac{c_{n+2}}{c_n} \tag{102}$$

for plaquette atmospheres and

$$\frac{\langle a_+^g \rangle_n}{\langle a_-^g \rangle_{n+1}} = \frac{c_{n+1}}{c_n} \tag{103}$$

for generalized atmospheres.

Since $[c_{n+2}/c_n] \to \mu^2$ (see equation (3)), the ratio of plaquette atmospheres approaches μ^2 as $n \to \infty$. More generally, the pattern theorem for walks and polygons [89, 90, 141] states shows that there exists constants α_+ and α_- such that to leading order

$$\langle a_+^p \rangle_n = \alpha_+ n + o(n), \tag{104}$$

$$\langle a_-^p \rangle_n = \alpha_- n + o(n), \tag{105}$$

and moreover, $[\alpha_+/\alpha_-] = \mu^2$.

Similar arguments can be made about the generalized atmospheric statistics for walks. In this case, since $[c_{n+2}/c_n] \to \mu^2$, it follows that

$$\left[\frac{\langle a_+^g \rangle_{n+1}}{\langle a_-^g \rangle_{n+2}}\right]\left[\frac{\langle a_+^g \rangle_n}{\langle a_-^g \rangle_{n+1}}\right] = \frac{c_{n+1}}{c_n} \cdot \frac{c_{n+2}}{c_{n+1}} \to \mu^2. \tag{106}$$

In the case of endpoint atmospheres, it similarly follows that $\langle a_+^e \rangle_n \langle a_+^e \rangle_{n+1} = [c_{n+2}/c_n] \to \mu^2$ as $n \to \infty$. In applications, one may assume that $[c_{n+1}/c_n] \to \mu$ and estimate μ [130].

3.2. Self-avoiding walk neutral atmospheres

A *neutral atmosphere* of a self-avoiding walk is a rearrangement of edges in the walk which do not change the length of the walk. As for positive atmospheres which increase length, or for negative atmospheres which decrease length, there are several possible definitions and we consider only a few here.

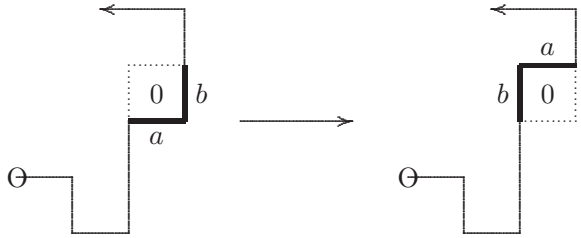

Figure 17. The plaquette marked with '0' is part of the plaquette neutral atmosphere of this walk. By reversing the order of the edges ab in the left walk to ba, the walk on the right is obtained. The total number of neutral plaquettes in the left walk is 4 and in the right walk is 2.

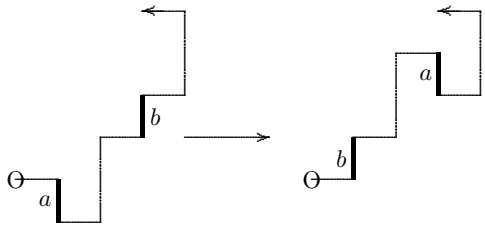

Figure 18. The edges marked a and b are part of the generalized exchange neutral atmosphere. By exchanging a and b in the walk on the left, the walk on the right is obtained. The total number of pairs of edges forming the generalized exchange neutral atmosphere is 35 in the walk on the left and 28 in the walk on the right.

(1) *Endpoint neutral atmospheres.* Suppose that s is a self-avoiding walk from the origin of length n and last edge e. A neutral endpoint atmosphere of s is defined by deleting e, rotating it into another orientation (possibly the same), and then reattaching it to s to obtain a new self-avoiding walk s'. This construction creates a linkage (s, s'). Observe that (s, s) is also a linkage. The number of endpoint atmospheres of s is denoted by $a_0^e(s)$. Since (s, s) is a linkage, $1 \leqslant a_0^e(s) \leqslant 2d - 1$ in d dimensions. If $a_0^e(s) = 1$, then s is a *trapped* conformation.

(2) *Plaquette neutral atmospheres.* Suppose that s is a self-avoiding walk and that a and b are two edges of s such that b follows a (that is ab is a subwalk of length two). If $a \perp b$, then exchanging a and b may create a new walk s' which has ba as a subwalk. This is illustrated for a particular walk in figure 17, and the reversal of the edges creates a linkage (s, s') between the two walks. Observe that (s, s) is not a linkage in this case. The number of plaquette neutral atmospheres of s is denoted by $a_0^p(s)$. There are walks s such that $a_0^p(s) = 0$, in this context.

(3) *Generalized exchange neutral atmospheres.* Suppose that s is a self-avoiding walk and that a and b are two edges in s. Then exchanging a and b may create a walk s', see figure 18, for an example. This exchange creates a linkage (s, s'). Since $s = s'$ whenever a is parallel to b (as vectors), (s, s) is also a linkage. The number of generalized exchanged atmospheres of s is denoted by $a_0^x(s)$, and $a_0^x(s) \geqslant 1$ for any walk of length longer than one.

(4) *Pivot neutral atmospheres.* Suppose that s is a self-avoiding walk and that v is a vertex in s. Then v cuts s in two subwalks w_0 and w_1 such that $w_0 v w_1 = s$. Without loss of

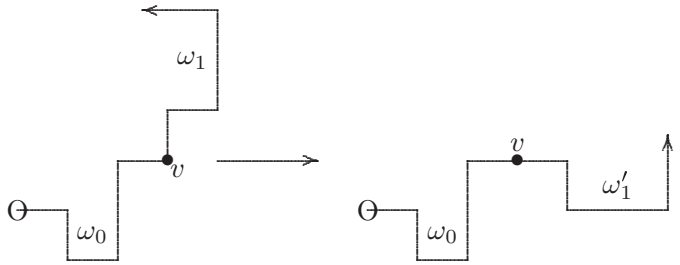

Figure 19. An example of a pivot neutral atmospheric subwalk. By rotating the subwalk s_1 starting at the vertex v denoted by • through an angle 90° clockwise, the subwalk s_1' on the right is obtained.

generality, suppose that w_1 is shorter than w_0. A *pivot neutral atmosphere* is defined by taking the shorter subwalk w_1, rotating and reflecting it by a symmetry operation which leaves the lattice invariant, into a subwalk w_1', to obtain the conformation $w_0 v w_1' = s'$. If s' is self-avoiding, then this defines a pivot neutral atmosphere. We give an example in figure 19. This construction is called a *pivot* and it creates a linkage (s, s'). Observe that (s, s) is also a linkage since the identity is also a symmetry operation leaving the lattice invariant. The number of pivot neutral atmospheres of s is denoted by $a_0^P(s)$ and $a_0^P(s) \geqslant 1$ for any walk.

In general, *neutral atmospheric moves* are defined by changing the conformation of a walk in its neutral atmospheres in a length-preserving way, as for example illustrated in figures 18 and 19. The examples above are by no means exhaustive.

3.3. Atmospheres and interacting models of walks

The atmospheres of self-avoiding walks define a set of models of walks with interacting atmospheres. These models are defined in analogy with models of collapsed walks as in figure 4: let $c_n(a_+, a_-, a_0)$ be the number or walks of length n, from the origin, with positive atmosphere of size a_+, negative atmosphere of size a_- and neutral atmosphere of size a_0. A given definition for the atmospheres can be used, and for each such definition, a different model will be obtained.

The partition function in this model is given by

$$Z_n(z_+, z_-, z_0) = \sum_{a_+, a_-, a_0} c_n(a_+, a_-, a_0) z_+^{a_+} z_-^{a_-} z_0^{a_0}. \tag{107}$$

This is a function of three parameters (z_+, z_-, z_0) and normally one would consider only one of the parameters at a time. For example, $Z_n(z_+, 1, 1) = Z_n^{a_+}(z_+)$ is the partition function on a model of walks with positive atmospheric statistic weighted by z_+.

In the case that endpoint atmospheric statistics are used in these models, one may show that

$$\lim_{n\to\infty} \frac{1}{n} \log Z_n(z_+, z_-, z_0) = \log \mu \tag{108}$$

independent of (z_+, z_-, z_0).

If plaquette atmospheric statistics are used, then one would expect a smooth crossover between walks with high values of the statistic to walks with low values of the statistic. Walks with low value of the statistic have a deficiency of plaquette atmospheres. While there is in

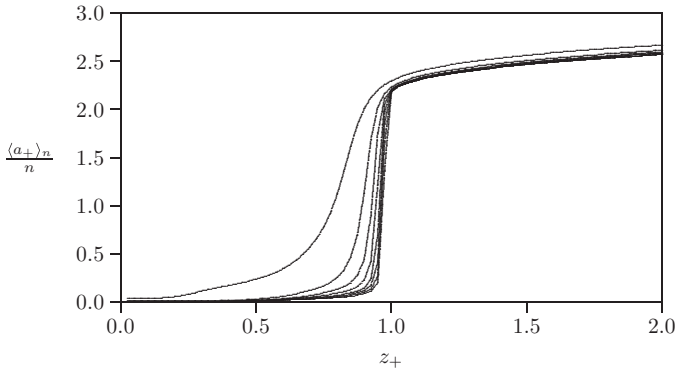

Figure 20. The average generalized positive atmosphere $\langle a_+\rangle_n$ per edge plotted against z_+ (and for $z_- = z_0 = 1$). The curves displayed are for $n = 30, 60, 90, \ldots, 240$. The left-most curve is data for $n = 30$, and the right-most curve is data for $n = 240$. There is a sharp transition from walks with small positive atmospheres to walks with large positive atmospheres close to $z_+ = 1$. These data are collected using the GARM algorithm in section 10.

this model no evidence of a transition, the limiting free energy is dependent on the parameters (z_+, z_-, z_0). Proving that the free energy exists for all values of (z_+, z_-, z_0) is an open question in this model as well.

For generalized atmospheric statistics this model is analogous to a model of collapsing walks. Consider first the model $Z_n(z_+, 1, 1)$. For small values of $z_+ < 1$ the walks in this model appear to be in a 'polygon phase': generally walks with small values of a_+^g have their endpoints close together as illustrated in figure 21.

It is not known that the free energy

$$\mathcal{F}_+^g(z_+) = \lim_{n\to\infty} \frac{1}{n} \log Z_n(z_+, 1, 1) \tag{109}$$

exists as a limit [3]. Numerical simulations show a sharp transition in this model at a critical value of z_+. Data for the average positive atmosphere per edge plotted against z_+ are displayed in figure 20. Similar data can be collected for the model $Z_n(1, z_-, 1)$.

The collapse phase in this model is also characterized by walks with small negative atmospheres, and they appear polygon-like as illustrated in figure 21(*a*). It is not known that the free energy

$$\mathcal{F}_-^g(z_-) = \lim_{n\to\infty} \frac{1}{n} \log Z_n(1, z_-, 1) \tag{110}$$

exists as a limit [3]. Similar to the positive generalized atmosphere case above, there is a sharp transition in this model.

Numerous other interacting models can be defined in this way by using an atmospheric statistic. In the generalized atmospheric statistic models, it appears that the collapsed phase is composed of walks with endpoints near each other, so that this phase may be called a 'polygon phase'.

4. Monte Carlo methods

Monte Carlo sampling of walks involves the statistical sampling of states from a distribution over a state space with the purpose of estimating expected values of observables computed

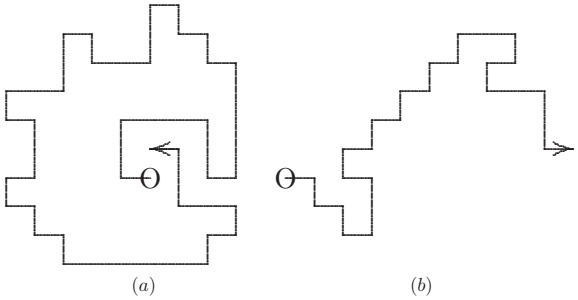

Figure 21. (*a*) A walk with generalized positive atmosphere $a_+^g = 0$. Generally walks with small positive atmospheres have endpoints closed together. Walks with small negative generalized atmospheres appear similarly to have their endpoints close together. This walk has $a_-^g = 2$. (*b*) A walk with large generalized positive atmosphere. These walks are more expanded than the example in (*a*) with small generalized positive atmosphere. A walk with a large negative generalized atmosphere also appears expanded as in this example. This walk has $a_+^g = 52$ and $a_-^g = 18$.

over the state space. The implementation of a Monte Carlo style sampling may take a variety of different forms, including direct sampling, kinetic-type algorithms or sampling from a distribution along a Markov Chain. In this context, any statistical sampling of walks will be loosely referred to as a Monte Carlo algorithm.

In the case of walks, the *state space S* can be defined in several ways, depending on the type and the implementation of the Monte Carlo algorithm. In some cases, *S* will be the abstract space of all conformations of walks, oriented from one endpoint and rooted in the origin. In other cases, it will be more appropriate not to orient or root the walks, but to consider equivalence classes of walks which are identical up to translation in the lattice.

The (normalized) distribution over a state space gives the probability that a given state will be sampled. In many cases this will not be the uniform distribution. For example, if the state space is infinite, then the probability of a given walk will normally depend on its length. An algorithm which samples walks (or any other object) of fixed size from the uniform distribution is a *canonical Monte Carlo algorithm*. If the algorithm samples walks of arbitrary size (length) from a distribution dependent on length, then the algorithm is a *grand canonical Monte Carlo algorithm*. Normally, the terms 'canonical' and 'grand canonical' refer to particular statistical ensembles of objects or states endowed with the Boltzmann distribution, but we abuse these terms to refer to ensembles of walks with fixed length (canonical) or arbitrary length (grand canonical).

Lastly, algorithms which sample directly from the state space *S* are called *static Monte Carlo algorithms*, while sampling along a Markov chain in the state space is a *dynamic Monte Carlo algorithm*.

4.1. Atmospheric moves and Monte Carlo algorithms

The detailed implementation of a particular Monte Carlo algorithm may be defined in terms of self-avoiding atmospheres and atmospheric moves. Consider the state space *S* of all self-avoiding walks. Positive and negative atmospheric moves, as well as neutral atmospheric moves, create linkages between walks. For example, the pair (s, s') may be a linkage where a positive atmospheric move changes s into s', and a negative atmospheric move changes s' into s. Neutral atmospheric moves similarly define linkages in the state space *S*.

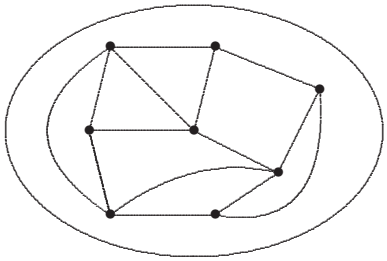

Figure 22. A graph of linkages in the state space of walks. Each vertex corresponds to a walk, each edge is a linkage.

The linkages created by atmospheric moves may be represented as a graph with vertices corresponding to walks (elements of the state space S), and edges corresponding to linkages. Such a graph is schematically illustrated in figure 22.

In the case of positive and negative atmospheres, the graph of linkages is an infinite graph with each vertex corresponding to a particular self-avoiding walk. If there is a path of finite length between any two vertices in the graph, then the corresponding walks may be changed into one another by applying a sequence of (positive and negative) atmospheric moves. If there is a path of finite length between any pair of vertices in the graph of linkages then we say that the corresponding set of positive and negative atmospheric moves is *irreducible*.

Since each positive atmospheric move can be reversed by a negative atmospheric move, a path between a walk s and a walk s' in the graph of linkages is reversible.

Similarly, neutral atmospheric moves create linkages in the state space of walks of fixed length n. This defines a finite graph of linkages with vertices which are walks of fixed length in the state space and edges the linkages defined by the neutral atmospheric moves. This situation is also schematically illustrated by figure 22, but in this case the graph of linkages is a finite graph.

We say that a set of neutral atmospheric moves is *irreducible* if its associated graph is connected. Similarly, the set is *reversible* if every path between two vertices can be reversed by the atmospheric moves.

Sets of atmospheric moves can be used to define Monte Carlo algorithms for sampling self-avoiding walks. If a set of neutral atmospheric moves is used, then the algorithm will sample walks of (given) fixed length, and the algorithm is a canonical Monte Carlo algorithm. If a set of positive and negative atmospheric moves is used to sample walks of arbitrary and variable length, then the algorithm is a grand canonical algorithm.

Generally, atmospheric moves and statistics will be a key factor in our approach to both the calculation of the properties of walks and in the design of algorithms for self-avoiding walks. For example, estimates of μ were obtained from atmospheric statistics in [130, 131], while endpoint atmospheric moves give rise to Rosenbluth-style sampling [132] of walks in the PERM [44] and flatPERM algorithms [5, 126], as well as the Berretti–Sokal algorithm [9] and under a slight generalization to the scanning method [105]. Similarly, plaquette atmospheres give rise to BFACF-style algorithms [8, 4], while generalized atmospheres give rise to GARM-style or GAS-style algorithms [131]. The pivot algorithm for walks of fixed length is defined by the implementation of pivot neutral atmospheric moves [92, 102]. From the most general point of view an implementation of a Monte Carlo algorithm proceeds by selecting a set of atmospheric moves and showing that the graph of linkages is connected. The implementation of the algorithm is normally via sampling along a Markov chain in the

graph of linkages. Analysing the properties of the chain gives information on the nature of the algorithm. In the following section, we discuss various implementation schemes.

4.2. Static Monte Carlo algorithms

In a static Monte Carlo algorithm independent states $\{s_1, s_2, \ldots, s_i, \ldots\}$ are sampled from a state space S such that state s_i is sampled with probability p_i. If the *weight* of state s_i is defined by $w_i = 1/p_i$, then the canonical average of an observable $\mathcal{O}$ is

$$\langle \mathcal{O} \rangle_N^{\text{est}} = \frac{\sum_{i=1}^N \mathcal{O}(s_i) w_i}{\sum_{i=1}^N w_i}. \tag{111}$$

For each realization of the states $\{s_1, s_2, \ldots, s_N\}$, the average $\langle \mathcal{O} \rangle_N^{\text{est}}$ is an independent estimate of the mean value $\langle \mathcal{O} \rangle$ over the entire state space S. The strong law of large numbers implies that $\langle \mathcal{O} \rangle_N^{\text{est}} \to \langle \mathcal{O} \rangle$ as $N \to \infty$ with probability 1. Moreover, since the averages $\langle \mathcal{O} \rangle_N^{\text{est}}$ are identically and independently distributed for any realization $\{s_1, s_2, \ldots, s_N\}$, the central limit theorem states that the estimated variance σ_N^{est} in the average $\langle \mathcal{O} \rangle_N$ is given by

$$\left[\sigma_N^{\text{est}}\right]^2 = \frac{\langle \mathcal{O}^2 \rangle_N^{\text{est}} - \left[\langle \mathcal{O} \rangle_N^{\text{est}}\right]^2}{N-1}. \tag{112}$$

That is, $\left(\langle \mathcal{O} \rangle_N^{\text{est}} - \langle \mathcal{O} \rangle\right) \big/ \sigma_N$ is asymptotically normally distributed with mean zero and variance one. A 67% statistical confidence interval on $\langle \mathcal{O} \rangle_N^{\text{est}}$ is estimated by σ_N^{est}, while $2\sigma_N^{\text{est}}$ is an estimated 95% confidence interval.

The implementation of a static Monte Carlo algorithm is in principle quite simple. States sampled from a distribution are independent and statistical analysis of quantities measured is simple. However, this does not mean that it is easy to code and use these algorithms, since there are other practical considerations which may make sampling inefficient, or impractical.

4.2.1. Approximate enumeration. Static Monte Carlo algorithms, including the Rosenbluth algorithm, as well as GARM, GAS and the flat versions of these algorithms, are approximate enumeration algorithms in the sense that the expected value of the weights w_i of walks of length n is equal to c_n [44, 59, 79, 131, 132] (the situation is slightly more complex in GAS in that a ratio of expected weights will be equal to c_n).

Hence, by estimating average weights in these algorithms, an estimate of c_n can be obtained; generally, one has

$$c_n = \langle w_i \rangle_n, \tag{113}$$

where $\langle \cdot \rangle_n$ indicates the expected value over all walks of length n.

Data obtained by approximate enumeration may be evaluated by using series analysis techniques, provided that the data are accurate enough to dampen out random noise and sampling biases. A starting point is the ratio estimator

$$r_n = \sqrt{\frac{c_{n+2}}{c_n}} \sim \left(1 + \frac{\gamma - 1}{n}\right) \mu \tag{114}$$

for μ which can be plotted against $1/n$. In approximate enumeration c_n in the above is replaced by approximate values obtained from a simulation. This was done in figure 23, where r_n was computed from approximations to c_n and then plotted against $1/n$. By extrapolating the ratio estimators to the intercept with the Y-axis and relying on equation (3), an estimate of μ is obtained. These particular estimators in figure 23 were generated by a simulation of flatGAS for $n \in [0, 249]$.

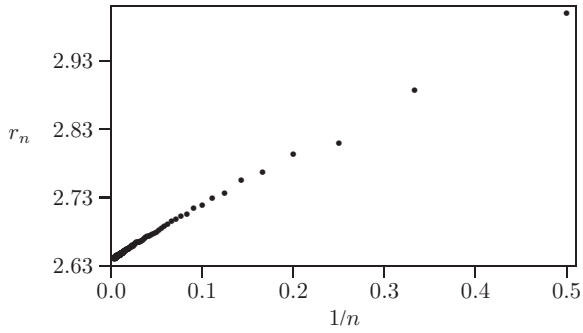

Figure 23. The ratio estimator r_n (see equation (114)) plotted against $1/n$ for two-dimensional data obtained by flatGAS for $n \in [2, 249]$. Extrapolating to 0 gives an estimate of μ.

A slightly better approach also involving ratio estimators is to use a linear extrapolant of r_n of the form

$$\mu_n = nr_n - (n-1)r_{n-1} \tag{115}$$

instead. The extrapolants μ_n scatter about μ, and averages or linear regressions may be used to determine best estimates of μ, often by ignoring data for small values of n.

Similarly, biased estimators for the entropic exponent γ are given by the extrapolant

$$\gamma_n = nr_n/\mu - n + 1, \tag{116}$$

where one may choose a best estimate for μ. See [79] for results on this using flatGAS for walks.

Other series techniques, including Padè and differential approximants, require very smooth data sets and have not been used successfully in modelling Monte Carlo data to extract critical exponents and growth constants. However, improved computing power and better approximate enumeration data may make it possible to apply more sophisticated series analysis techniques to Monte Carlo data, and this possibility remains an interesting and probable development in this field.

4.3. Dynamic Monte Carlo algorithms

Dynamic Monte Carlo algorithms involve certain dynamic rules (these are atmospheric moves which defines the 'elementary moves' of the algorithm) for sampling along a Markov chain in the state space S. The basic implementation is as follows: suppose that $\{s_1, s_2, \ldots, s_N\}$ is a realization of a Markov Chain in S. The next state s_{N+1} is obtained by applying an elementary move to s_N to propose the next state. The next state is accepted by applying a probabilistic rule. If the Markov Chain is ergodic (reversible and irreducible) and aperiodic, then the fundamental theorem of Markov chains implies that the sampling is from a unique stationary distribution, and statistical analysis of the sampled states can be done based on this.

4.3.1. Canonical dynamic Monte Carlo algorithms. This class of algorithms samples walks of fixed size from the uniform distribution. The algorithm is implemented via its microscopic dynamics: suppose that the sequence of walks $\{s_1, s_2, \ldots, s_N\}$ in state space S of walks of fixed length is a realization of a Markov chain. The next state s_{N+1} is obtained by applying an elementary move (which is a construction on s_N to change it into a newly proposed walk). We

assume here that the proposed state is generated with uniform probability by the elementary move. The proposed walk is accepted as s_{N+1} if it is a walk, otherwise it is rejected *and* $s_{N+1} = s_N$. That is, a state is read again into the Markov chain if the elementary move fails to produce a walk.

The microscopic dynamics of the algorithm set up a transition matrix P such that $P_{s\to t}$ is the probability of obtaining state t from state s by applying the elementary move. The elementary moves are independent of time, so that P is the transition matrix of a homogeneous Markov chain. Moreover,

$$\sum_t P_{s\to t} = 1 \tag{117}$$

since the chain will have a next state with probability 1. Thus P is a probability matrix with largest eigenvalue equal to 1.

If $\pi_s P_{s\to t} = \pi_t P_{t\to s}$ then P is *reversible* with respect to the distribution π (that is, the transition probabilities sample along the same Markov chain if time is reversed). The left eigenvector π defined by $\pi_s = \sum_t \pi_t P_{t\to s}$ is the stationary distribution of P if P is reversible with respect to π.

A canonical dynamic Monte Carlo algorithm is said to be *irreducible* if for any two given states s and t in the state space S, there is a realization of the Markov chain such that s and t are states in the chain. In other words, there is a finite value of n, such that $[P^n]_{st} > 0$.

The Monte Carlo algorithm is also said to be aperiodic if for any two given states s and t, there is an N such that $[P^n]_{s\to t} > 0$ for all $n \geqslant N$. If the algorithm is not aperiodic, then it may still be irreducible. If a Monte Carlo algorithm is both aperiodic and irreducible, then it is said to be *ergodic*.

A Markov chain Monte Carlo algorithm which is ergodic and reversible has a unique stationary distribution π. The distribution π is also its *limiting distribution* in the sense that $\rho_s[P^n]_{s\to t} \to \pi_t$ as $n \to \infty$ for any distribution ρ_s. That is, the algorithm samples states $s_j \in S$ asymptotically from the distribution π along a Markov chain irrespective of the initial state s_1. The Perron–Frobenius theorem guarantees the existence of the stationary distribution π, and also proves that $\lim_{n\to\infty} \rho_s[P^n]_{s\to t} = \pi$ for any initial distribution ρ. In this event, the algorithm is said to be *ergodic*.

If $\{s_1, s_2, \ldots, s_N\}$ is a realization of a Markov chain by the algorithm and $\mathcal{O}$ is an observable on the s_i, a natural estimator for the expected value $\langle\mathcal{O}\rangle$ is the average

$$\langle\mathcal{O}\rangle_N^{\text{est}} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{O}(s_i) \to \langle\mathcal{O}\rangle. \tag{118}$$

The convergence of $\langle\mathcal{O}\rangle_N^{\text{est}}$ to $\langle\mathcal{O}\rangle$ as $N \to \infty$ is a consequence of the ergodic theorem (see for example [42]).

The central limit theorem also holds, and the distribution of $\mathcal{O}(s_i)$ is asymptotically normal about $\langle\mathcal{O}\rangle$. The states s_i are correlated, and care must be taken in the analysis of variance of observables measured along the realized Markov chain.

4.3.2. Simulations of interacting models of walks by dynamic Monte Carlo simulations. Models of interacting walks, such as collapsing or adsorbing walks in figure 4, introduce a parameter $z = e^{\beta}$ in the model and define a partition function

$$Z_n(\beta) = \sum_{|s|=n} e^{\beta E(s)} = \sum_m c_n(m) e^{\beta m}, \tag{119}$$

where the first summation is over all walks s of length $|s| = n$ and energy $E(s)$, and where $c_n(m)$ is the number of walks of length n and *energy* m.

For example, in a model of walks adsorbing in a plane the energy m is the number of visits of vertices in the walk to the adsorbing plane. In a model of collapsing walks the energy m is the number of nearest neighbour contacts (vertices not adjacent in the walk but adjacent in the lattice) in the walk. Similarly, models of stretched and pulled walks have energies associated with each conformation. For small or negative values of β the partition function is dominated by walks with small values of the energy m. For large values of β the partition function is dominated by walks with large energies.

The simulation of interacting models of walks by a dynamic Monte Carlo algorithm poses unique numerical challenges. In the first instance, walks must be sampled along a Markov Chain from the Boltzmann distribution

$$P_\beta(s) = \frac{e^{\beta E(s)}}{Z_n(\beta)}. \tag{120}$$

Second, the mobility of the Markov chain may decrease for large values of β, increasing autocorrelation times along the chain and making it difficult to determine good numerical estimates of the mean values of observables. These sampling problems are normally referred to as *quasi-ergodicity* problems.

A symmetric canonical Monte Carlo algorithm can be used to sample from the distribution $P_\beta(s)$ by implementing it with the Metropolis algorithm [65, 110]. States proposed by the underlying elementary moves are accepted by a Metropolis criterion in such a way that the algorithm samples along a Markov chain from the Boltzmann distribution $P_\beta(s)$. The implementation of the Metropolis algorithm is as follows:

Algorithm 4.1 (Metropolis algorithm). Suppose that s_n is the current state, and that a proposal for a next state s' has been constructed from s_n using an elementary move in the underlying canonical Monte Carlo algorithm which samples states from the uniform distribution.

The following steps will find the next state s_{n+1} in the Markov chain:

(1) Construct a proposed next state s' with uniform probability by applying the elementary moves to s_n.
(2) Compute the ratio

$$q = \frac{P_\beta(s')}{P_\beta(s_n)},$$

where $P_\beta(s)$ is the Boltzmann distribution.
(3) The state s_{n+1} is now determined by the Metropolis criterion: if s' is a self-avoiding walk, then $s_{n+1} = s'$ with probability $\min\{q, 1\}$. Otherwise $s_{n+1} = s_n$.

The probability that a state t is obtained as the next state if s is the current state, $P_{s\to t}$, then satisfies the equation

$$P_\beta(s)P_{s\to t} = P_\beta(t)P_{t\to s} \tag{121}$$

and so $P_{s\to t}$ is reversible with respect to the Boltzmann distribution $P_\beta(s)$ over the state space S. By summing the above over s one sees that

$$\sum_s P_\beta(s)P_{s\to t} = P_\beta(t). \tag{122}$$

The rejection technique in the Metropolis algorithm implies that the algorithm samples along an aperiodic Markov chain. If the underlying elementary moves are irreducible, then the algorithm is ergodic, and it follows from equation (122) that it samples along a Markov chain asymptotically from the unique distribution $P_\beta(s)$. Observe that while we chose $P_\beta(s)$ to be

the Boltzmann distribution, that this is not required; one may substitute for it by any other distribution and then sample along a Markov chain. This observation underlies umbrella sampling [147], which we consider below.

Normally one is interested in estimating the expectation value of some observable $\mathcal{O}$ with respect to the Boltzmann distribution; that is

$$\langle \mathcal{O} \rangle_\beta = \frac{\sum_m c_n(m)\mathcal{O}(m)e^{\beta m}}{Z_n(\beta)}. \tag{123}$$

The natural estimator of this expectation value is the sample average

$$\langle \mathcal{O} \rangle_N^{\text{est}} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{O}(s_i), \tag{124}$$

where the algorithm realized the Markov Chain $s_1, s_2, \ldots, s_N$ by sampling N walks starting from the configuration s_1. The ergodic theorem applies here as well [42], and $\langle \mathcal{O} \rangle_N^{\text{est}} \to \langle \mathcal{O} \rangle_\beta$ almost surely as $N \to \infty$. The distribution of the observable $\mathcal{O}(s_i)$ is asymptotically normal about $\langle \mathcal{O} \rangle_\beta$, but are correlated, and analysis of variance must take this into account in determining a confidence interval on the sample mean.

4.3.3. Umbrella sampling. The problem with the Metropolis implementation of a canonical Monte Carlo algorithm for the simulation of interacting models of walks is that for large positive values of $z = e^\beta$ the Markov chain converges very slowly to its limit distribution. Typical realizations of the Markov chain spend long periods sampling small regions in state space and only slowly diffuse to other regions. This is the *quasi-ergodic problem* [147].

One approach to address this problem is to run the Metropolis algorithm at a smaller value of β, but then to use weighted averages to estimate sample means at larger values of β. This approach depends on the following identity: let π be a distribution different from P_β above. Then

$$\begin{aligned}
\langle \mathcal{O} \rangle_\beta &= \frac{\sum_m \mathcal{O}(m)c_n(m)P_\beta(m)}{\sum_m c_n(m)P_\beta(m)} \\
&= \frac{\sum_m \mathcal{O}(m)c_n(m)P_\beta(m)[\pi(m)/\pi(m)]}{\sum_m c_n(m)P_\beta(m)[\pi(m)/\pi(m)]} \frac{\sum_m \pi(m)}{\sum_m \pi(m)} \\
&= \frac{\sum_m \mathcal{O}(m)c_n(m)[P_\beta(m)/\pi(m)]\left[\pi(m)/\sum_m \pi(m)\right]}{\sum_m c_n(m)[P_\beta(m)/\pi(m)]\left[\pi(m)\Big/\sum_m \pi(m)\right]} \\
&= \frac{\langle \mathcal{O}P_\beta(m)/\pi \rangle_\pi}{\langle P_\beta(m)/\pi \rangle_\pi},
\end{aligned} \tag{125}$$

where $\langle \cdot \rangle_\beta$ is as before the expectation value with respect to the Boltzmann distribution P_β, while $\langle \cdot \rangle_\pi$ is the expectation value with respect to the distribution π. In other words, one may replace the Boltzmann distribution with π in the simulation, and then use the ratio of expectation values above to determine $\langle \mathcal{O} \rangle_\beta$ with respect to the Boltzmann distribution.

If $s_1, s_2, \ldots, s_N$ is the realization of a Markov chain from π, then the estimator for the canonical Boltzmann expectation value is given by

$$\langle \mathcal{O} \rangle_N^{\text{est}} = \frac{\sum_{i=1}^N \mathcal{O}(s_i)[P_\beta(s_i)/\pi(s_i)]}{\sum_{i=1}^N [P_\beta(s_i)/\pi(s_i)]}. \tag{126}$$

As before, $\langle \mathcal{O} \rangle_N^{\text{est}} \to \langle \mathcal{O} \rangle_\beta$ as $N \to \infty$.

This implementation of weighted averages works well if the distributions P_β and π overlap significantly. However, if the distribution P_β gives rise to quasi-ergodic problems, then π will similarly suffer from quasi-ergodic problems if it has significant overlap with P_β. This problem is overcome by umbrella sampling [147] and this relies on the fact that the distribution π can be chosen arbitrarily. If π is chosen such that it also has overlap with Boltzmann distributions where the algorithm has high mobility, then quasi-ergodic problems are largely avoided. Thus, the idea is to choose π carefully to be a wide distribution which overlaps both a Boltzmann distribution of interest and other Boltzmann distributions where the Markov chain converges fast. In this case π is called an *umbrella distribution*.

The flexibility in choosing π leaves one without an obvious set of criteria for choosing a suitable umbrella. Normally (see for example [145]) a weighted average of Boltzmann distributions can be chosen

$$\pi(s) = \sum_j w_j e^{\beta_j E(s)}, \tag{127}$$

where $E(s)$ is the energy of the state s, and where the set of parameters $\{\beta_1, \beta_2, \ldots, \beta_m\}$ includes values of β where the Markov chain is mobile as well as ranges of β where the Markov chain suffers from quasi-ergodic problems, but where we may want to compute sample averages. The weights w_j have to be chosen such that the individual Boltzmann distributions P_{β_j} contribute more or less equally to π. That is, the sampling should be approximately uniform in the parameter β over the entire range $[\beta_1, \beta_m]$.

This requirement can be satisfied by considering $\Pi = \sum_s \pi(s)$. This shows that

$$\Pi = \sum_s \pi(s) = \sum_s \sum_j w_j e^{\beta_j E(s)} = \sum_j w_j \sum_s e^{\beta_j E(s)} = \sum_j w_j Z_n(\beta_j). \tag{128}$$

The choice of the weights w_j in the above should be such that the contribution of the jth term to Π, given by $w_j Z_n(\beta_j)/\Pi$ is more or less a constant in j. That is, $w_j Z_n(\beta_j)$ should be independent of j, or only weakly dependent on it. Since the partition function is related to the finite n free energy $F_n(\beta)$ by $Z_n(\beta) = e^{nF_n(\beta)}$, this shows that a convenient choice for w_j is

$$w_j = Ce^{-nF_n(\beta_j)}, \tag{129}$$

where C is a constant. Thus, to use this method effectively one must have reasonable estimates of the free energy of the model. A practical implementation would first guess an umbrella and then use the results of initial simulations to determine estimates of the free energy which are used to update the umbrella. A few iterations of this process are enough to find a good umbrella distribution.

4.3.4. Multiple Markov chain Monte Carlo. Mobility in a Markov chain can also be improved by sampling along a set of Markov chains in parallel. This method is called multiple Markov chain Monte Carlo [41] and may be implemented efficiently in computers with parallel architecture. The method samples along Markov chains at different values of β simultaneously and swaps conformations between the chains; this improves mobility along any given chain.

Suppose that one wants to sample along a Markov chain at $z = e^\beta$ where convergence is slow, but that it is known that convergence is quick at a different value $z' = e^{\beta'} < e^\beta = z$. The idea is to select a sequence of values of β, say $\beta' = \beta_1 < \beta_2 < \cdots < \beta_m = \beta$, which interpolates between β and β'. β_j are chosen close enough together so that there is considerable overlap between adjacent Boltzmann distributions P_{β_j} and $P_{\beta_{j+1}}$.

The Markov chains at each of β_j are initiated and are sampled in parallel for a specified number of steps. Then an adjacent pair of chains at β_j and β_{j+1} is chosen uniformly from the

$m-1$ adjacent pairs, and as a trial move, the configurations at these β values are swapped. Suppose before the swap that the state s_j is in the chain at β_j, and state s_{j+1} is in the chain at β_{j+1}. If the distribution for given β is $P_\beta(s)$, then the swap is accepted with probability

$$r(j, j+1) = \min\left\{1, \frac{P_{\beta_{j+1}}(s_j)P_{\beta_j}(s_{j+1})}{P_{\beta_j}(s_j)P_{\beta_{j+1}}(s_{j+1})}\right\}. \tag{130}$$

The entire set of chains together with the swapping probability is itself a Markov chain which we call the *composite chain*.

Since each Markov chain is ergodic, the composite chain is ergodic, and its unique limit distribution is the product distribution of the separate Markov chains at β_j for $j = 1, 2, \ldots, m$. This follows immediately, since if $r(j, j+1) < 1$, then

$$\begin{aligned}\left[\prod_{k=1}^{m} P_{\beta_k}(s_k)\right]\frac{r(j, j+1)}{m-1} &= \left[\prod_{k=1}^{m} P_{\beta_k}(s_k)\right]\left[\frac{1}{m-1}\right]\left[\frac{P_{\beta_{j+1}}(s_j)P_{\beta_j}(s_{j+1})}{P_{\beta_j}(s_j)P_{\beta_{j+1}}(s_{j+1})}\right] \\ &= \left[\prod_{k\neq j, j+1} P_{\beta_k}(s_k)\right]\left[\frac{P_{\beta_{j+1}}(s_j)P_{\beta_j}(s_{j+1})}{m-1}\right]\end{aligned} \tag{131}$$

and similarly if $r(j, j+1) = 1$.

Every successful swap involving a given chain and its neighbour corresponds to a large change in the conformation of the state in each chain. This reduces autocorrelations along each chain. In addition, states in chains with low mobility also migrate to chains with high mobility; this alleviates quasi-ergodicity problems by providing new conformations in different parts of state space for low mobility chains.

Implementing multiple Markov chain Monte Carlo poses some practical considerations. First, a set of β values must be decided upon. If there are too many, then this slows down the sampling since CPU time increases with the number of chains. If there are too few, then there may not be enough overlap between adjacent chains and the swapping probability will be low. Thus, the number of chains, and their spacing, should be set such that a reasonable swapping probability is observed between each pair of chains. Second, the frequency of attempted swapping must be determined. By swapping pointers instead of conformations, swapping may be made computationally inexpensive, in which case swapping may be attempted frequently without increasing computational cost. See [77, 144, 145] for more details.

4.4. Grand canonical Monte Carlo algorithms

Grand canonical algorithms normally sample walks from a Boltzmann distribution over the lengths of the walks: the asymptotic probability that a walk s is observed along a realized Markov Chain is given by

$$P_t(s) = \frac{t^{|s|}}{\sum_s t^{|s|}}, \tag{132}$$

where $|s|$ is the length (or the number of edges) of the walk s.

The distribution (132) is only normalizable for small enough values of t, and the radius of convergence of $\sum_s P_t(s)$ defines a critical value t_c. If the parameter $t > t_c$, then the distribution P_t is not defined, and the grand canonical algorithm cannot be implemented.

Grand canonical algorithms are most easily implemented by sampling along a Markov chain using the Metropolis algorithm: elementary moves are constructed from positive and negative atmospheres of walk. A proposed state s' is constructed from the current state s_n using an elementary move. Once an elementary move is attempted, then the probability for

accepting s' as the next state in the Markov chain depends both on (1) the change in length and (2) whether s' is self-avoiding.

Algorithm 4.2 (Grand canonical sampling of walks). A metropolis implementation of grand canonical sampling of walks is as follows:

(1) Set the parameter t and let s_1 be an initial walk.
(2) Suppose the current state is s_n and apply an elementary move to propose a next state s'.
(3) If s' is not a self-avoiding walk, then reject it and put $s_{n+1} = s_n$.
(4) If s' is a self-avoiding walk, then determine the ratio

$$q = \frac{P_t(s')}{P_t(s_n)} = t^{(|s'|-|s_n|)}.$$

Put $s_{n+1} = s'$ with probability $\min\{q, 1\}$, otherwise put $s_{n+1} = s_n$.
(5) If the requisite number of iterations has been performed, then terminate the algorithm, otherwise continue at step (2).

If the initial state is s_1, then this implementation realizes a Markov chain $\{s_1, s_2, s_3, \ldots, s_N\}$. The mean length (number of edges) in the walks s_n depends on the parameter t, and the states along the chain are correlated so that analysis of variance of observables must be carried out with some care. Observe that aperiodicity is built in this algorithm by the rejection technique, but that irreducibility of the elementary moves must be proven.

The probability $P_{s\to r}$ that a state r will be observed to follow state s along the chain is given by the product of the probability $p(s)$ that r will be proposed by the elementary move if the current state is s, and by the conditional probability that r will be accepted if proposed given by $\min\{1, q\}$. Thus

$$P_{s\to r} = p(s)\min\{1, q\}. \tag{133}$$

Similarly, one may determine $P_{r\to s}$, and it follows that the detailed balance condition in these grand canonical algorithms are given by

$$\frac{P_t(s)P_{s\to r}}{p(s)} = \frac{P_t(r)P_{r\to s}}{p(r)}. \tag{134}$$

Thus, the algorithm is reversible with respect to the distribution $[P_\beta(s)/p(s)]$. If the algorithm is irreducible, then by summing both sides of equation (134) over s (and noting that $\sum_s P_{r\to s} = 1$), it follows that

$$\sum_s P_{s\to r}\left[\frac{P_t(s)}{p(s)}\right] = \left[\frac{P_t(r)}{p(r)}\right]. \tag{135}$$

In other words, the algorithm samples asymptotically from the invariant distribution given by

$$D_t(s) = \frac{[P_t(s)/p(s)]}{\sum_r [P_t(r)/p(r)]}. \tag{136}$$

In implementations where $p(r) = p(s)$ this reduces to the Boltzmann distribution in equation (132). Normally, $p(s)$ will be related to the length of the walk s: for example, in the BFACF-algorithm [4, 8] it is the case that $p(s) = 1/(2(d-1)|s|)$ in d dimensions, in which case the detailed balance condition in equation (134) is given by $|s|t^{|s|}P_{s\to r} = |r|t^{|r|}P_{r\to s}$, and the algorithm samples asymptotically from the distribution $A|s|t^{|s|}$ over the state space S, with A being a normalization constant.

Sampling along a Markov chain with a grand canonical algorithm from the Boltzmann distribution (132) with $t < t_c$ gives a correlated Markov chain of realized states which we

denote by $\{s_1, s_2, s_3, \ldots, s_N\}$. Each state is a walk of length $|s_i|$, and the initial state s_1 is an arbitrary walk used to initialize the sampling.

The asymptotic distribution for walks of fixed given length or size along the chain is uniform, and the (canonical) average of an observable $\mathcal{O}$ can be determined by sieving states of given length from the chain and then computing $\langle\mathcal{O}\rangle_N$ as in equation (118). The strong law of large numbers applies in this case as well, while the central limit theorem implies that the distribution of $\langle\mathcal{O}\rangle_N$ for fixed size is asymptotically normal about $\langle\mathcal{O}\rangle$. This is not a very efficient way of sampling walks of given size, but in some applications may still be the best way to proceed [71, 74].

Alternatively, the (grand canonical) average

$$\langle\mathcal{O}\rangle_{t,N}^{\mathrm{est}} = \frac{1}{N}\sum_{i=0}^{N}\mathcal{O}(s_i) \tag{137}$$

at given value of the activity t can be analysed. In this case, the law of large numbers shows that $\langle\mathcal{O}\rangle_{t,N}^{\mathrm{est}}$ converges to its mean value

$$\langle\mathcal{O}\rangle_t = \sum_r D_t(r)\mathcal{O}(r). \tag{138}$$

In the case that $N \to \infty$, $\langle\mathcal{O}\rangle_{t,N}^{\mathrm{est}} \to \langle\mathcal{O}\rangle_t$ almost surely as $N \to \infty$.

Canonical Boltzmann averages can be determined by determining averages of $p(s)\mathcal{O}(s)$ and $p(s)$ instead. Then the ratio estimator

$$\frac{\langle p(s)\mathcal{O}\rangle_{t,N}}{\langle p(s)\rangle_{t,N}} \to \langle\mathcal{O}\rangle_t^{\mathrm{can}} = \sum_r P_t(r)\mathcal{O}(r) \tag{139}$$

converges to the canonical average over the Boltzmann distribution as $N \to \infty$.

In grand canonical algorithms, the correlations between states along the realized Markov chain can be very long (and is typically more persistent than correlations present in the canonical algorithms).

4.5. Canonical and grand canonical Monte Carlo algorithms for interacting models

The most general implementation of the Metropolis algorithm is in models of interacting walks. In such models each walk s is endowed with an *energy* $E(s)$, and the goal is to use a Monte Carlo algorithm to sample from the Boltzmann distribution

$$P_\beta(s) = \frac{e^{\beta E(s)}}{\sum_r e^{\beta E(r)}}, \tag{140}$$

where β can be interpreted as an 'inverse temperature'. Comparison to equation (56) shows that $z = e^\beta$ in this model, and the general comments in section 2.3 apply here.

If the sampling is canonical (in a state space of fixed length walks) then this distribution is normalizable for any value of β. On the other hand, if the sampling is grand canonical, then $E(s)$ may depend in some way on the length of the walk s, and in which case there is a critical value β_c of β.

In the case of canonical sampling of walks of fixed length, the implementation of the Metropolis algorithm follows the steps outlined in algorithm 4.1. If s_n is the current state, then an elementary move is applied to construct a proposed state s'. The parameter q is given by

$$q = e^{\beta(E(s')-E(s_n))} \tag{141}$$

and s' is accepted as the next state in the Markov chain with probability $\min\{1, q\}$.

In the event that a grand canonical simulation is done, elementary moves may change the length of the walk, in which case the edges are weighted by t. If s_n is the current state, and a proposed state s' is constructed by an elementary move. Then parameter q is given by

$$q = t^{(|s'|-|s_n|)} e^{\beta(E(s')-E(s_n))} \tag{142}$$

and s' is accepted as the next state in the Markov chain with probability $\min\{1, q\}$. In other words, in a grand canonical implementation the 'energy' of a state s has a term proportional to its length and is given by $|s|\log t + \beta E(s)$. There are two parameters (t, β) in this case. Normally, t is fixed at a convenient value so that walks of sufficient length are generated, while β is changed to explore the effects of the energy term. The βt-plane is a phase space for the model, with a critical line $t_c(\beta)$ which determines the radius of convergence of the generating function (see equations (58) and (59)). In this event, one identifies the limiting free energy of the model, given by $\mathcal{F}(\kappa) = -\log t_c(\kappa)$ as in equation (59), where $z = e^{\beta}$.

The same observations made for interacting models apply in this case. As above, the algorithm samples along a Markov chain asymptotically from the invariant limit distribution of the algorithm is given by

$$D(s) = \frac{[P_\beta(s)/p(s)]}{\sum_r [P_\beta(r)/p(r)]} \tag{143}$$

for given t and β. This reduces to a Boltzmann distribution if the probability of selecting an elementary move $p(s)$ is independent of s.

Given a realization of a Markov chain $\{s_1, s_2, s_3, \ldots, s_N\}$ in a canonical implementation of the algorithm will give the estimator

$$\langle \mathcal{O} \rangle^{\text{est}}_{\beta,N} = \frac{1}{N} \sum_{i=0}^{N} \mathcal{O}(s_i) \tag{144}$$

of the mean value

$$\langle \mathcal{O} \rangle_\beta = \sum_r D(r)\mathcal{O}(r). \tag{145}$$

As above, in the case that $p(s)$ is not independent of s, canonical averages can be determined by using the ratio estimator in equation (139).

4.6. Analysis of variance

Static algorithms, such as the Rosenbluth, PERM and GARM, produce independently sampled walks. If the walks are not sampled from the uniform distribution, then weighted averages must be determined, and averages are computed as in equation (111). The variance σ_N in equation (112) gives a 67% confidence interval on the computed average.

The situation is more complicated for dynamic Monte Carlo algorithms such as the Berretti–Sokal and pivot algorithms. For both canonical and grand canonical sampling realizing a Markov chain $\{s_1, s_2, \ldots, s_N\}$ of states, the observables along the sequence of states, given by $\{\mathcal{O}(s_1), \mathcal{O}(s_2), \ldots, \mathcal{O}(s_N)\}$, are correlated. The average value of this correlated sequence is

$$\langle \mathcal{O} \rangle^{\text{est}}_{N} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{O}(s_i) \tag{146}$$

and it is asymptotically an unbiased estimator distributed normally about the expected value $\langle \mathcal{O} \rangle$. In figure 24, a time series for the square radius of gyration of square lattice walks of length $n = 200$ plotted as a function of iterations of the pivot algorithm. The data are correlated

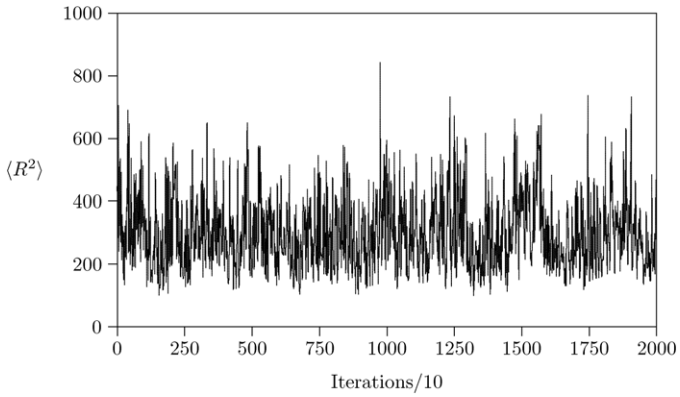

Figure 24. Time series data from a simulation of self-avoiding walks of length $n = 200$ using the pivot algorithm. Plotted is the square radius of gyration against the iterations. The data are correlated, and time series analysis can be used to determine means and variances.

and computing averages requires careful analysis of the autocorrelation times along this time series.

The variance of the distribution of $\langle \mathcal{O} \rangle_N^{\text{est}}$ about the mean is harder to compute. Although the time series of measured observables $\{\mathcal{O}(s_i)\}$ has identically distributed elements, they are not independent, and autocorrelations must be calculated to determine confidence intervals.

The dependence of an observable along a time series is statistically measured by an autocorrelation function. Normally, the autocorrelation function decays at an exponential rate measured by the autocorrelation time $\tau_{\mathcal{O}}$ along the time series. The *'connected' autocorrelation function*

$$S_{\mathcal{O}}(k) = \langle \mathcal{O}(s_i)\mathcal{O}(s_{i+k}) \rangle - \langle \mathcal{O} \rangle^2 \tag{147}$$

along the Markov chain measures the correlations between states a distance of k steps apart along the time series. The time series is asymptotically homogeneous (independent of its starting point); this implies that $\langle \mathcal{O}(s_i) \rangle = \langle \mathcal{O} \rangle$ independent of s_i. In other words, the autocorrelation time $S_{\mathcal{O}}(k)$ is only dependent on the separation k between the states $\mathcal{O}(s_i)$ and $\mathcal{O}(s_{i+k})$. If the states are uncorrelated, then the chain is homogeneous and $\langle \mathcal{O}(s_i)\mathcal{O}(s_{i+k}) \rangle$ factors. Thus

$$\langle \mathcal{O}(s_i)\mathcal{O}(s_{i+k}) \rangle = \langle \mathcal{O} \rangle^2. \tag{148}$$

In this case $S_{\mathcal{O}}(k) = 0$ if $k > 0$ and $S_{\mathcal{O}}(0)$ is the variance of the observable $\mathcal{O}$.

More generally, the sequence $\{\mathcal{O}(s_i)\}$ is an asymptotically homogeneous time series of correlated states each with identical distribution but correlated by the Markov property to prior states. In this case, the autocorrelation function is given by

$$\begin{aligned} S_{\mathcal{O}}(k) &= \langle \mathcal{O}(s_i)\mathcal{O}(s_{i+k}) \rangle - \langle \mathcal{O} \rangle^2 \\ &= \langle (\mathcal{O}(s_i) - \langle \mathcal{O} \rangle)\,(\mathcal{O}(s_{i+k}) - \langle \mathcal{O} \rangle) \rangle\,. \end{aligned} \tag{149}$$

With increasing values of k the autocorrelation function $S_{\mathcal{O}}(k)$ decays to leading order at an exponential rate $\tau_{\mathcal{O}}$ which is called the *exponential autocorrelation time*:

$$S_{\mathcal{O}}(k) \simeq C_{\mathcal{O}} e^{-k/\tau_{\exp,\mathcal{O}}}. \tag{150}$$

The exponential autocorrelation time of the observable $\mathcal{O}$ is then defined by

$$\tau_{\exp,\mathcal{O}}^{-1} = \limsup_{k\to\infty} \left[\frac{-\log S_{\mathcal{O}}(k)}{k} \right]. \tag{151}$$

The exponential autocorrelation time of the time series (and thus of the underlying Monte Carlo dynamics) is defined by

$$\tau_{\text{exp}} = \sup_{\mathcal{O}} \tau_{\text{exp},\mathcal{O}} \tag{152}$$

and it gives the timescale of the slowest modes (or longest autocorrelations) in the simulation.

An effective algorithm will have a short exponential autocorrelation time. As a matter of principle, this also gives the timescale of the decay of initial bias in the algorithm. In other words, if $\{s_1, s_2, \ldots, s_N, \ldots\}$ is a realization of a Markov chain with initial state s_1, then the bias introduced by the choice of a particular s_1 can be assumed to have been lost in statistical noise after $m\tau_{\text{exp}}$ iterations of the elementary moves of the algorithm, for large enough values of m.

As a practical matter it is not possible to compute the exponential autocorrelation time. Instead, the autocorrelation time is estimated for a particular observable $\mathcal{O}$ and then used to determine the variance and a confidence interval on averages of $\mathcal{O}$ measured along the time series.

Summing equation (150) shows that

$$2\sum_{k=1}^{\infty} \frac{S_{\mathcal{O}}(|k|)}{S_{\mathcal{O}}(0)} \simeq (2\tau_{\text{int},\mathcal{O}} - 1), \tag{153}$$

where $\tau_{\text{int},\mathcal{O}}$ is the *integrated autocorrelation time* of the observable $\mathcal{O}$. Solving for $\tau_{\text{int},\mathcal{O}}$ shows that

$$\tau_{\text{int},\mathcal{O}} = \frac{1}{2} + \sum_{k=1}^{\infty} \frac{S_{\mathcal{O}}(k)}{S_{\mathcal{O}}(0)}. \tag{154}$$

If the time series is uncorrelated, then $S_{\mathcal{O}}(k) = 0$ whenever $k \neq 0$, and the result is that $\tau_{\text{int},\mathcal{O}} = 1/2$ in that case. If the times series is correlated, then $\tau_{\text{int},\mathcal{O}} > 1/2$. This definition assumes that the autocorrelation function decays as a pure exponential. This is not strictly the case, since there may be more than one exponential mode in the time series, each with its own characteristic autocorrelation time. In other words, for the observable $\mathcal{O}$ we assume that $\tau_{\text{int},\mathcal{O}}$ is the *longest* or dominant autocorrelation time, describing the slowest or most persistent correlations in the stream. This assumption is normally not true, since there may be very slow modes in the time series which are masked by noise and by larger, but more quickly decaying, autocorrelations. However, for any given observable $\mathcal{O}$, calculating $\tau_{\text{int},\mathcal{O}}$ is assumed to be sufficient for setting statistical confidence intervals.

The variance of an observable $\mathcal{O}$ over the time series $\{s_1, s_2, \ldots, s_N\}$ can now be computed if one assumes that N (the length of the time series) is very large:

$$\begin{aligned}
\left[\sigma_{\mathcal{O}}^{\text{est}}\right]^2 &= \left\langle \left(\mathcal{O} - \langle\mathcal{O}\rangle_N^{\text{est}}\right)^2 \right\rangle_N^{\text{est}} \\
&= \frac{1}{N^2} \left\langle \sum_{i,j=1}^{N} (\mathcal{O}(s_i) - \langle\mathcal{O}\rangle_N)(\mathcal{O}(s_j) - \langle\mathcal{O}\rangle_N) \right\rangle_N^{\text{est}} \\
&= \frac{1}{N^2} \sum_{i,j=1}^{N} S_{\mathcal{O}}(|i-j|) \\
&\approx \frac{2S_{\mathcal{O}}(0)}{N} \left(\frac{1}{2} + \sum_{k=1}^{\infty} \frac{S_{\mathcal{O}}(k)}{S_{\mathcal{O}}(0)} \right) \\
&= \left[\frac{2\tau_{\text{int},\mathcal{O}}}{N} \right] S_{\mathcal{O}}(0).
\end{aligned} \tag{155}$$

If the data are uncorrelated, then $\tau_{\mathrm{int},\mathcal{O}} = 1/2$ and the variance reduces to the usual expression. For correlated data, this shows that the number of effectively independent measurements is $N/[2\tau_{\mathrm{int},\mathcal{O}}]$.

4.6.1. Computing confidence intervals: I. Statistical confidence intervals can be computed without directly measuring $\tau_{\mathrm{int},\mathcal{O}}$. A realization of a (correlated) time series $\{s_1, s_2, \ldots, s_N\}$ by a Monte Carlo algorithm is analysed by cutting it into blocks of size M, and then treating the blocks as independent measurements of an observable $\mathcal{O}$.

Define the average of $\mathcal{O}$ over the ith block by

$$[\mathcal{O}]_i = \frac{1}{M}\sum_{j=1}^{M} \mathcal{O}(s_{iM+j}) \tag{156}$$

for $i = 0, 1, 2, \ldots, \lfloor N/M \rfloor$. Then $\{[\mathcal{O}]_i\}$ is itself a time series and its variance can be computed by

$$s^2_{M,\mathcal{O}} = \langle [\mathcal{O}]^2 \rangle - \langle [\mathcal{O}] \rangle^2. \tag{157}$$

If $[\mathcal{O}]_i$ are treated as independent measurements of $\langle \mathcal{O} \rangle$, then the confidence interval $\sigma_{M,\mathcal{O}}$ on the average is given by

$$\sigma^2_{M,\mathcal{O}} = \frac{s^2_{M,\mathcal{O}}}{\lfloor N/M \rfloor - 1} \tag{158}$$

since there are $\lfloor N/M \rfloor$ measurements $[\mathcal{O}]_i$ of $\langle \mathcal{O} \rangle$.

If $M = 1$, then each observable is a block of size 1, and since the data are correlated, $\sigma_{1,\mathcal{O}}$ is an underestimate of the confidence interval. Increasing the block size M also increases $\sigma_{M,\mathcal{O}}$, and for M comparable to $2\tau_{\mathrm{int},\mathcal{O}}$ the blocks should become statistically independent. At this point, $\sigma_{M,\mathcal{O}}$ becomes stable, or only weakly dependent on M, and can be taken as a statistical confidence interval. This should occur, for large values of N, when $M \sim 2\tau_{\mathrm{int},\mathcal{O}}$.

4.6.2. Computing confidence intervals: II. A second method for estimating statistical confidence intervals involves the calculation of integrated autocorrelation times. A 'windowing' scheme [102] provides a convenient method for doing this. Suppose that $\{s_1, s_2, \ldots, s_N\}$ is a Markov chain of states realized by a dynamic Monte Carlo algorithm. Compute the average of an observable $\mathcal{O}$ over this realization:

$$\langle \mathcal{O} \rangle_N = \frac{1}{N}\sum_{i=1}^{N} \mathcal{O}(s_i). \tag{159}$$

The autocorrelation function is estimated by

$$S_{\mathcal{O}}(k) \approx \frac{1}{N-k}\sum_{i=1}^{N-k} \left(\mathcal{O}(s_i)\mathcal{O}(s_{i+k}) - \langle \mathcal{O} \rangle^2_N \right). \tag{160}$$

Since $S_{\mathcal{O}}(k) \sim e^{-k/\tau_{\mathrm{int},\mathcal{O}}}$, the autocorrelation time should decay with increasing k, and when $k > M > m\tau_{\mathrm{int},\mathcal{O}}$ for m large (say $m \approx 50$), then the estimate for $S_{\mathcal{O}}(k)$ should be less than statistical noise in the time series.

If k is cut off at M, then the estimate for $\tau_{\mathrm{int},\mathcal{O}}$ is

$$\tau_M = \frac{1}{2} + \sum_{k=1}^{M} \frac{S_{\mathcal{O}}(k)}{S_{\mathcal{O}}(0)}. \tag{161}$$

Implement this estimate recursively as follows:

Algorithm 4.3 (Estimating $\tau_{\mathrm{int},\mathcal{O}}$).

(1) Put $\tau_m = 1/2$ for each $m \in \{2, 5, 10, 25, 50, 100\}$ proceed to step (2).
(2) Determine τ'_m by summing

$$\tau'_m = \frac{1}{2} + \sum_{k=1}^{m\tau} \frac{S_{\mathcal{O}}(k)}{S_{\mathcal{O}}(0)} \tag{162}$$

for each value of m.
(3) If $|\tau_m - \tau'_m| < \epsilon$ for some small ϵ, then τ_m is the estimate of $\tau_{\mathrm{int},\mathcal{O}}$ at window $M = m\tau$. Otherwise, put $\tau_m = \tau'_m$ and recursively update τ by executing step (2) above for each m.
(4) With increasing values of the window size m, τ_m will stabilize. This value is the estimate of $\tau_{\mathrm{int},\mathcal{O}}$.

This algorithm only works well if $N \gg M = m\tau_{\mathrm{int},\mathcal{O}}$ and m typically greater than 20. In other words, the window should be much larger than the autocorrelation time, and much smaller than the length of the time series. If either of these conditions fails, then it may be the case that an erroneous estimate of the integrated autocorrelation time was computed.

Generally, the estimation of autocorrelation times is fraud with difficulties which may conspire to give unreliable results. The only real cure for such difficulties is a time series which is sufficiently long compared to the correlations which may be present. Thus, an effective sampling scheme (algorithm) and long computer simulations give the best results. Generally, good results are obtained if the time series contains at least $1000\tau_{\mathrm{int},\mathcal{O}}$ states for the measurement of the observable $\mathcal{O}$.

4.6.3. Initialization bias. Initialization bias in the average $\langle\mathcal{O}\rangle_N$ of an observable $\mathcal{O}$ due to a choice of a slowly mixing initial state over a time series of length N can be avoided by discarding the first M states of the Markov chain for sufficiently large values of M. That is, $\langle\mathcal{O}\rangle_N$ is computed as a function of M, until it shows no dependence on M apart from statistical noise.

In principle, the length scale of initialization bias is set by the exponential autocorrelation time τ_{exp}, and this may be much longer than the integrated autocorrelation times used in determining statistical confidence intervals. In this respect, initialization bias introduces a systematic error. Practical considerations show that this bias can be removed by throwing away the first 25τ measurements along the time series, but this is only a rule of thumb, and data should be examined to make sure that such biases are not present.

5. Simple sampling

Simple sampling of a self-avoiding walk is the simplest algorithm for generating walks of small length. It is a very inefficient manner of sampling, but improvements which are not obvious give rise to more efficient sampling algorithms, for example, the Rosenbluth algorithm [132] in section 6.

Simple sampling of self-avoiding walks proceeds by choosing a starting vertex v_0 at the origin, and then appending an edge to it in a direction selected uniformly to obtain the next vertex v_1. The vertex v_n is obtained recursively by adding an edge (selected uniformly from one of $2d - 1$ possible directions) to v_{n-1}. If a self-intersection occurs then the conformation is thrown away, and a new walk is initiated from the origin.

Algorithm 5.1 (Simple sampling). The algorithm samples a collection of independent walks of length n from the uniform distribution and can be implemented by the following steps:

(1) Let v_0 be the vertex at the origin, and determine v_1 by choosing one of the $2d$ nearest neighbour vertices to v_0.
(2) Determine v_m recursively by choosing one of the $2d-1$ nearest neighbours of v_{m-1} in the lattice.
(3) If v_m intersects a previously selected vertex, then reject the entire walk and start a new walk at step (1).
(4) If $m < n$, increment m and continue at step (2). If $m = n$, then a walk of desired length has been generated; start the next walk from step (1) and continue until a desired number of walks have been sampled.

Simple sampling is an algorithm which samples uniformly from the state space of random lattice walks without a back step (that is, without stepping back over the last step). These walks are filtered and the self-avoiding walks are retained.

The walks generated by simple sampling are independent, and simple averages of measured quantities are unbiased estimators of the properties of walks of given length. This algorithm is also very inefficient. Attrition of walks is exponential in the length n, and few walks of significant length will be sampled even if many iterations are attempted.

If simple sampling produces a collection of N walks of length n, say $\{s_1, s_2, \ldots, s_N\}$, then an unbiased estimator of the observable $\mathcal{O}$ is the average

$$\langle \mathcal{O} \rangle = \frac{1}{N} \sum_{i=1}^{N} \mathcal{O}(s)_i. \tag{163}$$

The walks are statistically independent and a confidence interval is given by the standard deviation $\sigma_{\mathcal{O}}$ defined by

$$\sigma_{\mathcal{O}}^2 = \frac{\langle \mathcal{O}^2 \rangle - \langle \mathcal{O} \rangle^2}{N-1}. \tag{164}$$

The algorithm is rarely used, but is relevant because improvements to it give efficient sampling schemes.

6. Rosenbluth sampling of self-avoiding walks

Rosenbluth sampling of self-avoiding walks is an improvement on simple sampling. The implementation is similar, but is modified by choosing the next vertex in a walk only from those which are not already occupied. The implementation is as follows: a starting vertex is chosen at the origin, and the next vertex is selected recursively from the set of nearest neighbours which are not already occupied by the walk; these vertices are in the endpoint atmosphere of the walk. If there are no nearest neighbours available, then the entire (incomplete) walk is rejected and a new walk is started at the origin.

The algorithm generates a set of N walks $\{s_i\}$ of length n of weights W_i.

Algorithm 6.1 (Rosenbluth sampling).

(1) Put $W = 1$ and let v_0 be the vertex at the origin, and determine v_1 by choosing one of the $2d$ nearest neighbour vertices to v_0.
(2) Suppose that v_{m-1} has σ_{m-1} nearest neighbour vertices which are not already occupied. Then $\sigma_{m-1} = a_+^e$, where a_+^e is the endpoint atmosphere of the walk $v_0 v_1 \cdots v_{m-1}$. Choose v_m recursively by choosing a vertex from the unoccupied neighbours of v_{m-1} with probability $1/\sigma_{m-1}$.

(3) If $\sigma_m = 0$ then the walk is trapped, and it is rejected. A new walk is started at step (1).
(4) Update the weight $W \to W\sigma_{m-1}$.
(5) If $m < n$, then determine the next step by going to step (2). If $m = n$ then a walk of length n and weight W has been generated. Continue by starting a new walk at step (1) until N walks are generated.

The Rosenbluth algorithm does not generate a sample of self-avoiding walks of length n from the uniform distribution over state space. Instead, the probability $P(s)$ of a walk s of length n and weight $W(s)$ being sampled is equal to $1/W(s)$ (the inverse of the weight computed by the algorithm), and where $W(s)$ is given by

$$W(s) = \prod_{i=0}^{m-1} \sigma_i(s) \tag{165}$$

and where $\sigma_i(s)$ is the number of open nearest neighbours of the ith vertex in the partially grown walk of length $i-1$ prefixing s. Observe that walks which have $\sigma_i = 0$ for some i are technically included in the sample of walks, but with zero weight.

The partition function of walks of length n is

$$Z_n = \sum_{|s|=n} P(s)W(s) \tag{166}$$

exactly if the sum is over all walks of length n. Since $P(s) = 1/W(s)$, it follows that $Z_n = c_n$, where c_n is the number of walk of length n from the origin.

If a simulation of self-avoiding walks using Rosenbluth sampling produces a collection of N walks of length n, say $(s_1, s_2, \ldots, s_N)$ and with associated weights $(W(s_j))$, then the estimate of the partition function is

$$\langle Z_n \rangle_N = \frac{1}{N} \sum_{i=1}^{N} W(s_i). \tag{167}$$

Since $Z_n \equiv c_n$, where c_n is the number of walks of length n, it follows from the strong law of large numbers that $\langle Z_n \rangle_N \to c_n$ as $N \to \infty$. In other words, the Rosenbluth algorithm is basically a method for the approximate enumeration of walks.

An estimator of the observable $\mathcal{O}$ over a sample of N walks is the average

$$\langle \mathcal{O} \rangle_N^{\text{est}} = \frac{\sum_{i=1}^{N} W(s_i)\mathcal{O}(s_i)}{\sum_{i=1}^{N} W(s_i)}. \tag{168}$$

Walks with zero weight are included in the above ratio estimator, but with zero weight. Moreover, by the strong law of large numbers $\langle \mathcal{O} \rangle_N^{\text{est}} \to \langle \mathcal{O} \rangle$ as $N \to \infty$. In addition, the central limit theorem applies and the estimates $\langle \mathcal{O} \rangle_N$ are asymptotically normally distributed about the expected value. Since walks are also statistically independent, a confidence interval is given by the standard deviation $\sigma_{\mathcal{O}}$ defined by

$$\sigma_{\mathcal{O}}^2 = \frac{\langle \mathcal{O}^2 \rangle_N - \langle \mathcal{O} \rangle_N^2}{N-1}. \tag{169}$$

Rosenbluth sampling is a significant improvement over simple sampling, and longer walks can be sampled efficiently. Attrition of walks by trapped conformations becomes a serious problem in this algorithm when the length of walks exceeds about 150 steps in the square lattice. There is less attrition in higher dimensions, but even then attrition compounds quickly with increasing length. In the square and cubic lattices, the dispersion of weights W_i of walks of length $n \gtrapprox 100$ increases quickly so that they are spread over many orders of

magnitude. Eventually a few walks with very large weights will dominate statistical averages and effectively reduce the sample size. This causes the variance of the sample to increase and at this point the algorithm fails to give reliable results.

The Rosenbluth method can be adapted to sample walks in an interacting model from a Boltzmann distribution

$$P_\beta = \sum_s e^{\beta E(s)}, \tag{170}$$

where $E(s)$ is the energy of the walk s. In this case, the algorithm is modified by computing the change in E for each of the nearest neighbours of the last vertex v_{m-1}. Suppose that the change in E for the kth nearest neighbour is $\Delta E_{m-1}^{(k)}$. Then the kth nearest neighbour is chosen as the next vertex with probability

$$u_m^{(k)} = \frac{e^{\beta \Delta E_{m-1}^{(k)}}}{\sum_{j=1}^{s_m} e^{\beta \Delta E_{m-1}^{(j)}}}. \tag{171}$$

In d dimensions v_{m-1} has $2d$ nearest neighbours, and one can deal with self-intersections by putting $\Delta E_{m-1}^{(k)} = -\infty$ whenever the kth nearest neighbour is occupied by vertices already sampled. The probability of a given walk s of energy $E(s)$ is

$$P(s) = \prod_{m=1}^{n} u_m^{(k_m)} = \frac{e^{\beta E(s) - E_0)}}{\prod_{m=1}^{|s|} \sum_{j=1}^{s_m} e^{\beta \Delta E_{m-1}^{(j)}}}, \tag{172}$$

where (k_m) is the sequence of nearest neighbours realized by the algorithm. Canonical averages are computed by reweighing the walks appropriately: suppose that $\langle s_j \rangle$ has been sampled by the algorithm, and that the weight of a walk s is given by

$$W(s) = \prod_{m=1}^{|s|} \sum_{j=1}^{s_m} e^{\beta \Delta E_{m-1}^{(j)}}. \tag{173}$$

The partition function is similarly given (up to a constant) by

$$Z_n(\beta) = \sum_s P(s) W(s) = e^{-\beta E_0} \sum_s e^{\beta E_s} \tag{174}$$

and thus the estimate of the average weights in a simulation is given by

$$\langle Z_n(\beta) \rangle_N = \frac{1}{N} \sum_{i=1}^{N} W(s_i) \tag{175}$$

if a sequence of N walks $\langle s_1, s_2, \ldots, s_N \rangle$ was generated by the algorithm. It follows that $\langle Z_n(\beta) \rangle_N \to Z_n(\beta)$ as $N \to \infty$ so that this algorithm is a method for estimating the partition function. Expected values of observables are similarly computed by

$$\langle \mathcal{O} \rangle_{\beta,N}^{\text{est}} = \frac{\sum_{i=1}^{N} W(s_i) \mathcal{O}(s_i)}{\sum_{i=1}^{N} W(s_i)} \tag{176}$$

is an estimator for the mean of $\mathcal{O}$ at fugacity β. A confidence interval can be estimated similarly.

7. Dimerization of self-avoiding walks

Dimerization [1] is a process whereby the construction of self-avoiding walks is attempted by concatenating shorter walks into longer walks. The method proceeds by generating a

set of N walks and then dimerizing them in pairs. Each resulting walk is examined for self-intersections (between its two halves), and both are removed from the collection if they intersect. The remaining dimerized walks are kept and may be further dimerized in subsequent iterations of the algorithm.

The method is implemented as follows:

Algorithm 7.1 (Dimerization).

(1) Generate, by using another Monte Carlo method, a set S_0 of N_0 *independent* walks of length n.
(2) Suppose that S_m is a set of N_m walks of length $2^m n$ edges. Dimerize pairs of walks in S_m; if the resulting walk is self-avoiding (of length $2^{m+1}n$), then add it to the set S_{m+1}. Otherwise, reject both walks and attempt a dimerization of two new walks selected from S_m. Continue until there is only one walk or fewer remaining in S_m.
(3) If there are more than M walks in S_{m+1}, then increment m and continue at step (2). Otherwise terminate the algorithm, and S_m is a collection of at least M self-avoiding walks of length $2^m n$.

In this implementation, the algorithm will generate a collection of at least M independent walks, of length $2^m n$, where m is the number of successive dimerizations of the walk. The walks are contained in sets $S_0, S_1, \ldots, S_m$. It is important to note that while all the walks in set S_k are independent of one another, the sets S_k and S_{k+1} are not independent: walks in S_{k+1} were constructed from pairs of walks in S_k.

The probability of a successful dimerization attempt of a given pair of walks can be approximated as follows: two walks starting in the origin will not intersect with probability approximately c_{2n}/c_n^2. Substituting $c_n \approx An^{\gamma-1}\mu^n$ into this gives $Cn^{1-\gamma}$ for some constant C. Since $\gamma > 1$ in low dimensions, this argument shows that the probability that two given independent walks will dimerize decreases as a power law with their length.

Hence, if $|S_0|$ is the size of a set of independent walks of length n, then after dimerization a set S_1 of approximately $|S_1| \approx |S_0|n^{(1-\gamma)}/2$ walks of length $2n$ is obtained. In three dimensions, $\gamma \approx 1.16$ so that $1 - \gamma \approx 1/6$, and $|S_1|/|S_0| \sim n^{-1/6}$. Data in [1] support this rough estimate. The algorithm is not as efficient in two dimensions.

An acceleration procedure for dimerized walks is described in [2]: by noting that two walks may be dimerized in four different ways and biasing the dimerization in favour of 'swollen' conformations, the attrition of walks is reduced. Dimerization was also used to extend the exact series of self-avoiding walks [104].

8. The scanning method

The scanning method [105] is a generalization of simple sampling and of the Rosenbluth method. The algorithm selects steps one at a time to create a walk, but at each step scans possible future walks to determine the probabilities for selecting a given step. In this way it samples walks of given length from the uniform distribution.

The method is initiated by starting a walk at the origin in the d-dimensional hypercubic lattice, and then giving a first step in one of $2d$ directions with probability $1/2d$. Suppose that a walk of length m with steps (edges) $\sigma_1, \sigma_2, \ldots, \sigma_{m-1}$ has been generated. The endpoint atmosphere a_+^e of the last vertex t_{m-1} gives the number of open vertices next to t_{m-1} available for giving the next step σ_m. The probability for selecting σ_m from amongst the a_+^e possibilities is dependent on all the previous steps $\sigma_1, \sigma_2, \ldots, \sigma_{m-1}$ of

the walk, and we denote it by the conditional probability $p_m(\sigma \mid \sigma_1, \sigma_2, \ldots, \sigma_{m-1}) = \Pr[\sigma \text{ is the next step} \mid \text{the previous steps were } \sigma_1, \sigma_2, \ldots, \sigma_{m-1}]$.

The scanning method is implemented by exactly computing the conditional probability $p_m(\sigma \mid \sigma_1, \sigma_2, \ldots, \sigma_{m-1})$ at each step, further conditioned on the fact that the final walk must have length n. Thus, the probability of the mth step is computed by taking into account all the partial walks of $n - m + 1$ steps which will complete the current partially generated walk of length $m - 1$ into a walk of length n.

The number of partial walks of length $n - m + 1$ which can complete the walk $\sigma_1, \sigma_2, \ldots, \sigma_{m-1}$ into a walk of length n, given that the mth step is σ, is denoted by $N_n(\sigma \mid \sigma_1, \sigma_2, \ldots, \sigma_{m-1})$. For each possible mth step σ, this number will vary and it is determined by an exact enumeration procedure. We define the (conditional) transition probability for the mth step in the direction σ to be given by

$$p_m(\sigma \mid \sigma_1, \sigma_2, \ldots, \sigma_{m-1}) = \frac{N_n(\sigma \mid \sigma_1, \sigma_2, \ldots, \sigma_{m-1})}{\sum_\sigma N_n(\sigma \mid \sigma_1, \sigma_2, \ldots, \sigma_{m-1})} \tag{177}$$

and the mth step is given in direction σ by stepping with probability $p_m(\sigma \mid \sigma_1, \sigma_2, \ldots, \sigma_{m-1})$. Once a walk of length n is constructed, then a new walk is started from scratch, and eventually a collection of independent walks is generated.

The probability of obtaining a given walk $s = \sigma_1, \sigma_2, \ldots, \sigma_n$ is given by

$$P_s = \frac{1}{2d} \prod_{m=2}^{n} p_m(\sigma_m \mid \sigma_1, \sigma_2, \ldots, \sigma_{m-1}). \tag{178}$$

Since the algorithm enumerates all walks up to length n at each step of the algorithm, each walk is generated with the same probability, and so $P_s = c_n^{-1}$.

The scanning method is implemented by the following steps:

Algorithm 8.1 (The scanning method).

(1) Initiate a walk by giving a first step σ_1 in one of $2d$ directions from the origin in the hypercubic lattice.
(2) Suppose that the walk s of length $m - 1$ with steps $\sigma_1, \sigma_2, \ldots, \sigma_{m-1}$ has been generated. Determine the number of walks $N_n(\sigma \mid \sigma_1, \sigma_2, \ldots, \sigma_{m-1})$ of length $n - m + 1$ and with first step σ which can complete s into a walk of length n by scanning (enumerating) the future steps of s. This is done for σ taking $2d - 1$ different directions.
(3) Choose a direction for σ with probability

$$p_m(\sigma|\sigma_1, \sigma_2, \ldots, \sigma_{m-1}) = \frac{N_n(\sigma|\sigma_1, \sigma_2, \ldots, \sigma_{m-1})}{\sum_\sigma N_n(\sigma|\sigma_1, \sigma_2, \ldots, \sigma_{m-1})}.$$

(4) Increment m by one. If $m = n$ then a walk of length n has been obtained, otherwise, $m < n$ and the algorithm continues from step (2).

The underlying operation of the scanning method relies on determining generalized endpoint atmospheres of length $n - m + 1$ for a partially generated walk, and then stepping uniformly into each atmospheric partial walk. Generally, this is a computationally expensive algorithm for generating self-avoiding walks; for $m = 2, 3, \ldots, n$ walks of length $n - m + 1$ must be counted, and this shows that the computational effort is at least given by the effort for enumerating all walks of length n. By abandoning rigour however and settling for approximate enumeration of only say b steps ahead at each iteration, the algorithm can be improved significantly. This modification makes the algorithm more efficient, at the expense of uniformity: walks are not sampled with uniform probability any more.

The modified algorithm is implemented in the same way as the scanning method, but instead of scanning ahead $n - m + 1$ steps at each iteration of the algorithm, only b steps are scanned ahead (more precisely, only $\min\{b, n - m + 1\}$ are scanned ahead at each step). The parameter b is chosen small enough to make the algorithm fast and large enough to give a distribution over the walks which is approaching uniformity.

In this implementation, the number of walks $N_n(\sigma, b \mid \sigma_1, \sigma_2, \ldots, \sigma_{m-1})$ of length b which can be appended to a current walk of length $m - 1$ with the first step given in the σ-direction is determined for each σ. The probability of giving the next step σ is then

$$p_m(\sigma, b \mid \sigma_1, \sigma_2, \ldots, \sigma_{m-1}) = \frac{N_n(\sigma, b \mid \sigma_1, \sigma_2, \ldots, \sigma_{m-1})}{\sum_\sigma N_n(\sigma, b \mid \sigma_1, \sigma_2, \ldots, \sigma_{m-1})} \tag{179}$$

and the probability of a given walk s is

$$P_s = \frac{1}{2d} \prod_{m=2}^{n} p_m(\sigma_m, b \mid \sigma_1, \sigma_2, \ldots, \sigma_{m-1}). \tag{180}$$

Since not every given walk can be completed to a walk of length n, the result is that the probabilities P_s are not normalized over walks of length n, but $\sum_s P_s < 1$. This must be taken into account.

Averages can be computed by assigning a weight $1/P_s$ to each walk, and then an observable $\mathcal{O}$ is approximated by

$$\langle \mathcal{O} \rangle_N = \frac{\sum_{i=1}^{N} \mathcal{O}(s_i)/P_{s_i}}{\sum_{i=1}^{N} 1/P_{s_i}} \tag{181}$$

if the walks $\{s_i\}_{i=1}^N$ were sampled, with walk s_i obtained with probability P_{s_i}. This approximation depends on b and P_s.

Increasing b generally improves the numerical performance of the algorithm (at the expense of CPU time). If b is taken equal to $n - m + 1$ at the mth step, then the original scanning algorithm is uncovered with the P_s uniform so that the last expression reduces to a simple average. Taking $b = 1$ at each step uncovers a method related to the Rosenbluth method. The larger b is, the flatter the distribution over the walks sampled, while less walks are lost to attrition since they cannot be completed into walks of length n. This improves the estimate $\langle \mathcal{O} \rangle_N$ systematically, and by tracking this as b is increased, one can determine the uncertainties in approximate estimates of observables.

In implementations of the scanning method, b is usually increased to be at most equal to 10, and then the approximations are extrapolated to infinite b. This procedure generally gives good results [106, 108], and the method may yet again be resurrected for further refinement and applications in walks.

9. Pruned enriched Rosenbluth sampling (PERM)

The Rosenbluth algorithm suffers two basic flaws: the first is the attrition of walk due to trapped conformations;' this is a problem even for walks of length 100 in low dimensions. The second flaw is even more serious: the dispersion of the weights of the generated walks increases to the point that a few walks (or even a single walk) start to dominate the sample averages. This dispersion of the weights fatally undermines the effectiveness of Rosenbluth sampling.

Both these flaws can be overcome by the introduction of pruning and enrichment steps in Rosenbluth sampling [44]. Walks with large weights are enriched (see [153]) in the sampling while having their individual weights reduced, and walks with low weights are pruned. These

measures reduce the dispersion of the weights, and (by enrichment) increase the number of longer walks in the simulation. The enrichment and pruning steps added to the Rosenbluth algorithm gives PERM, an acronym for 'pruned and enriched Rosenbluth method'.

In simple sampling, enrichment is implemented by adding copies of surviving walks at a fixed rate. Suppose that N walks are started in the simple sampling method, and that the rate of attrition (due to trapped conformations) is α; that is, the number of surviving conformations at length n is $S_N(n) \simeq Ne^{-\alpha n}$. To compensate for this attrition, double up each surviving chain after every k iterations so that after the first k iterations, the number of surviving conformations is $S_N(k) \simeq 2Ne^{-\alpha k}$.

Thus, after n-iterations, the number of surviving walks is

$$S_N(n) \simeq N2^{\lfloor n/k \rfloor} e^{-\alpha n}. \tag{182}$$

Since we wish to keep the number of walks roughly constant by this enrichment strategy, one should choose k such that

$$\frac{\lfloor n/k \rfloor \log 2}{n} \to \alpha > 0 \tag{183}$$

as $n \to \infty$. Thus, $k \approx \lfloor \log 2/\alpha \rfloor$ would be an acceptable choice. The value of α can be measured empirically and then k can be chosen.

In the case of Rosenbluth sampling, attrition is also a serious problem; but this is compounded by the increase in the dispersion of the weights. Enrichment can be used to both reduce large weights and to decrease attrition of the sample size for longer walks.

Suppose that the weight of a conformation s of length n is given by

$$W(s) = \prod_{i=1}^{n-1} \sigma_i(s). \tag{184}$$

Introduce a cut-off weight or threshold T_n at length n, and if $W(s) > T_n$, then enrich the sample by a adding a copy of s to it, while at the same time reducing the weight $W(s)$ by a factor of 2. In other words, there are now two copies of s, but each with a weight of $W(s)/2$. While this enrichment does not affect the sample average at n, continued growth of the walks from this value of n will result in the copies growing along different trajectories, and their enrichment has the effect of reducing the dispersion.

The problem of walks with small weights is dealt with by introducing a lower threshold t_n at length n. If a walk is grown by the Rosenbluth algorithm to length n and of weight $W(s)$, and $W(s) < t_n$, then the walk is pruned (its growth is stopped and it is assigned zero weight) with a probability $1/q$ where q is a parameter of the algorithm. If the walk is not pruned (with probability $1 - 1/q$), then its weight is increased by a factor of q. Normally, $q = 2$.

Sample averages are computed as before. If N is the number of walks that were started, then sample averages are computed over the entire sample of pruned and enriched conformations as in equation (168), where the weights of walks which were pruned are put equal to zero.

Algorithm 9.1 (PERM).

(1) Let T_m and t_m be thresholds on the weights of a walk of length m sampled by the Rosenbluth algorithm. Set the value of $q > 1$ for pruning and an integer $k > 1$ for enrichment.
(2) Put $W = 1$ and let v_0 be the vertex at the origin, and determine v_1 by choosing one of the $2d$ nearest neighbour vertices in its positive endpoint atmosphere (of size a_+^e) to v_0.
(3) Suppose that v_{m-1} has positive endpoint atmosphere of size σ_{m-1} nearest neighbour vertices which are not already occupied. Choose v_m recursively by choosing a vertex from the unoccupied neighbours of v_{m-1} with probability $1/\sigma_{m-1}$.
(4) If $\sigma_{m-1} = 0$ then the walk is trapped and it is rejected. A new walk is started at step (1).

(5) Put $W = W\sigma_{m-1}$ and increment n.
(6) If $m = n$, then a walk of length n and weight W is found, start the next walk at step (1).
(7) If $m < n$ and $W < t_m$, then with probability $1/q$ put $W = 0$ and start a new walk at step (1). Otherwise put $W \to qW$. Continue this walk by starting at step (2).
(8) If $m < n$ and $W > T_m$, then create k copies of the walk, each of weight W/k. Grow each copy to length n by continuing the algorithm at step (2).
(9) Start at step (1) until N walks have been started.

The thresholds t_n and T_n can be adapted during the simulation. Normally, one would start with $t_n = 0$ and T_n large. During the simulation t_n and T_n are changed while requiring T_n/t_n to stay roughly constant, say $T_n/t_n \approx 10$. One way in which this can be implemented is to use the partition function Z_n: suppose that a collection of N walks of length n, say $(s_1, s_2, \ldots, s_N)$ and weights $(W(s_j))$ have been sampled. Then the estimate of the partition function is

$$\langle Z_n \rangle_N = \frac{1}{N} \sum_{j=1}^{N} W(s_j) \tag{185}$$

and $\langle Z_n \rangle_N \to c_n$ as $N \to \infty$. The thresholds t_n and T_n are then chosen by $t_n = c\,\langle Z_n \rangle_N$ and $T_n = C\langle Z_n \rangle_N$, where c and C are fixed so that $C/c \approx 10$.

Approximate enumeration of walks by PERM is performed by taking averages over the weights of walks of length n:

$$c_n^{\text{est}} = \langle W_N \rangle_n \approx \frac{1}{N} \sum_{j=1}^{N} W(s_j). \tag{186}$$

Estimates of mean values of observables are computed over the weighted sample of walks by

$$\langle \mathcal{O} \rangle_n^{\text{est}} = \frac{\sum_{j=1}^{N} \mathcal{O}(s_j) W(s_j)}{\sum_{j=1}^{N} W(s_j)}. \tag{187}$$

This algorithm has been used effectively to sample self-avoiding walks of impressive lengths [44].

PERM can be implemented to sample walks from a Boltzmann distribution. In this case, the walks should be weighted so that the partition function

$$\langle Z_n \rangle_{\beta,N} = \frac{1}{N} \sum_{i=1}^{N} W(s_i) e^{\beta E(s_i)} \to \langle W(s) e^{\beta E(s)} \rangle \tag{188}$$

is obtained as $N \to \infty$. Walks can be grown either as before, and then having the Boltzmann factor attached as part of its weight, or it can be grown using importance sampling: if there are m_n nearest neighbours available for the next step, each of energy E_i for $i = 1, 2, \ldots, m_n$, then step to vertex i with probability $p_n = e^{\beta E_i} \big/ \sum_{j=1}^{m_n} e^{\beta E_j}$. The weights are then computed as before, and averages are taken consistently with these weights.

9.1. Flat-histogram pruned enriched Rosenbluth sampling (flatPERM)

The average weight $\langle W \rangle_N$ in Rosenbluth sampling is an estimate of the total number of walks c_n. If fluctuations in this estimate are suppressed, then improved estimates of c_n can be obtained.

In general, Rosenbluth sampling has large fluctuations for larger values of n, and so cannot be used to accurately measure c_n. PERM is one method for reducing fluctuations.

A further refinement in PERM can be achieved by favouring copies of walks with large weights to grow in different ways. This is achieved as follows: let $\langle W\rangle_N$ be the estimate of c_n after N walks have been sampled. If the Nth walk has weight W_N, then compute

$$r = \frac{W_N}{\langle W\rangle_N}. \tag{189}$$

If $r > 1$ then the Nth walk has a larger than expected weight, and it should be enriched. If $r > a_n$ (where a_n is the number of possible ways of extending the walk by adding a step), and $c = \min\{\lfloor r\rfloor, a_n\}$, then the walk is enriched in the sample by making c copies of it. If $r < 1$, then its weight is smaller than expected, and it can be pruned. This is implemented as follows:

Algorithm 9.2 (flatPERM).

(1) Implement Rosenbluth sampling and suppose that N walks of length n and of weights W_N have been sampled.
(2) Compute $r = W_N/\langle W\rangle_N$.
(3) Suppose that $r > 1$: put $c = \min\{\lfloor r\rfloor, s_n\}$. Make c copies of the last walk each of weight W_N/c.
(4) If $r \leqslant 1$ then prune: continue growing with probability r and weight W_N/r, and prune with probability $1 - r$.
(5) Continue by growing new walks from step (1).

Observe that the pruning and enriching are done *after* the current conformation is included in computing $\langle W\rangle_N$. The estimate $\langle W\rangle_N$ for c_n is initially very wrong, but it improves with increasing N. In this implementation, the number of states (walks) at each length n is roughly constant—this gives a flat histogram of the number of walks sampled at each value of n.

Implementation of flatPERM produces a roughly constant number of walks at each value of n: for each walk pruned on average, one walk is enriched. This gives a significant improvement over Rosenbluth sampling, where attrition makes the sampling of long walks very difficult. In figure 25, the attrition of walks in Rosenbluth sampling is measured for a million started walks in the square lattice, with maximum possible length set to 1000. All walks started in this simulation terminated in a trapped conformation at lengths shorter than 1000, and only a few walks (fewer than 25) of lengths longer than 500 were sampled.

Adding pruning and enrichment steps in a flatPERM implementation of a million started walks of lengths up to 1000 gives a dramatic improvement, as shown in figure 25. In this case the attrition is less than 10% even for $n = 1000$ and the histogram is more or less flat over the entire range of $n \in [0, 1000]$. This gives a large sample of long walks, compared to the Rosenbluth sampling.

Implementations of flatPERM require more CPU time for a given number of started walks, compared to the Rosenbluth algorithm. This occurs because more walks are completed. The flatPERM data in figure 25 required roughly 20 times the CPU time of the Rosenbluth data presented there.

Normally a flatPERM simulation will start with poor estimates of the average weights $\langle W\rangle$. This affects the initial effectiveness of pruning and enrichment, leading to small decreases in completed walks. This difficulty can be dealt with by not growing long walks initially: that is, restrict the length of the first walks until statistics have built up to give better estimates of the weights, and at this point the restriction can be removed to generate longer walks. One particular scheme is to limit the length of walks to $n < cS$ where S is the number of walks generated. As the algorithm runs, S increases until n reaches its desired size at which point the restriction is removed. This reduces the number of grown walk to roughly $\lfloor S - n/c\rfloor$. The constant c is typically fixed at values between 1 and 10.

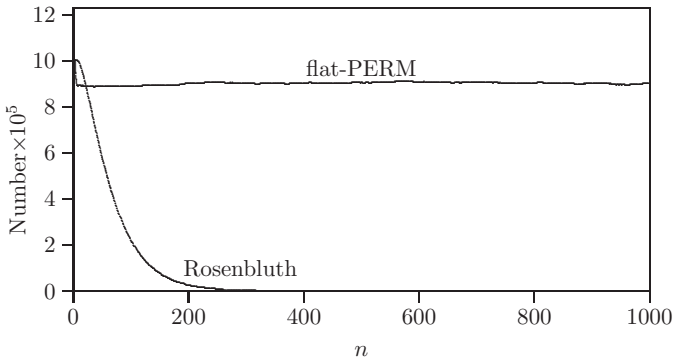

Figure 25. Attrition of started walks in the Rosenbluth and flatPERM algorithms. The curves above correspond to the number of surviving walks of length n. This quickly decreases to zero in the Rosenbluth algorithm. In the case of the flatPERM algorithm, a virtually constant number of walks is obtained after initial attrition due to thermalization of the algorithm. In these simulations a million walks were started in each simulation, and the number of surviving walks at each value of n is measured. The area under each curve is proportional to the total number of steps into positive endpoint atmospheres given by the algorithms and also proportional to the CPU time of each simulation. In this particular example, the CPU time of the flatPERM simulation is 20 times longer than the CPU time of the Rosenbluth simulation.

The thermal implementation of flatPERM proceeds by assuming that at length n a sample of N walks has been generated with estimated partition function

$$\langle Z_n \rangle_{\beta,N} = \langle W e^{\beta E} \rangle_N. \tag{190}$$

The threshold is

$$r_n = \frac{W_n e^{\beta E_n}}{\langle Z_n \rangle_{\beta,N}} \tag{191}$$

and the implementation of enrichment and pruning is done as before. The threshold criterion is that each term $W_n e^{\beta E_n}$ gives the same size contribution to the partition function, this gives a flat distribution over n for any fixed choice of β.

9.2. Microcanonical implementation of flatPERM

A microcanonical implementation of flatPERM will produce estimators of $c_n(m) \equiv c_{n,m}$; this is the number of walks of length n and energy m. In this incarnation of the algorithm, weights $W_{n,m}$ are computed as before but by tracking the energy as edges are added: the average weight of walks is then computed for each value of m; this gives an estimate of $c_{n,m}$:

$$c_{n,m}^{\text{est}} = \langle W_{n,m} \rangle = \frac{1}{N} \sum_i W_{n,m}(s_i), \tag{192}$$

where $W_{n,m}(s_i) = 0$ if $E(s_i) \neq m$. In other words, like canonical flatPERM, this algorithm is primarily an approximate enumeration algorithm, designed to estimate the numbers $c_{n,m}$.

Pruning and enrichment proceed as in algorithm 9.2 but now with r replaced by

$$r = \frac{W_{n,m}}{\langle W_{n,m} \rangle} \tag{193}$$

at the ith step.

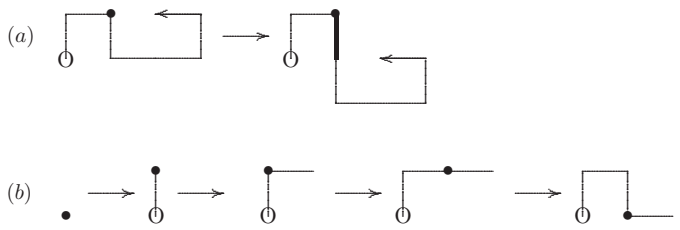

Figure 26. (*a*) Inserting an edge into the positive generalized atmosphere of a walk. The vertex marked with a • in the walk on the left-hand side has two generalized atmospheric edges incident with it; one pointing in the south direction, and the other pointing in the east direction. By inserting an edge in the south direction at this vertex, the walk on the right-hand side is obtained. On the other hand, the bold edge in the walk on the right-hand side is a negative atmospheric edge in the walk. By contracting it, the walk on the left-hand side is obtained. The walk on the left-hand side has generalized atmospheric statistics $a_+^g = 14$ and $a_-^g = 3$, and the walk on the right has atmospheric statistics $a_+^g = 16$ and $a_-^g = 4$. (*b*) Sampling generalized positive atmospheres in GARM. Starting from the trivial walk, at each iteration a vertex • is chosen and a positive atmospheric edge is inserted to generate the next state in the sequence.

In the microcanonical implementation the algorithm performs a random walk in n and m, and while correlations between enriched walks still build up, there is roughly a constant number of conformations for each size n and energy m. In this sense, the algorithm produces a 'flat' histogram in (n, m); see [91] for an implementation in a model of pulled adsorbing walks as an example.

Averages of an observable $\mathcal{O}$ are determined as before by the ratios

$$\langle \mathcal{O} \rangle_{n,m}^{\text{est}} = \frac{\sum_{(i)} \mathcal{O}_{n,m}^{(i)} W_{n,m}^{(i)}}{\sum_{(i)} W_{n,m}^{(i)}}. \tag{194}$$

The microcanonical implementation of this algorithm gives a good method to determine canonical averages at a given fugacity β:

$$\langle \mathcal{O} \rangle_{n}^{\text{est}}(\beta) = \frac{\sum_m \mathcal{O}_{n,m}^{\text{est}} C_{n,m}^{\text{est}} e^{\beta m}}{\sum_m C_{n,m}^{\text{est}} e^{\beta m}}. \tag{195}$$

10. The generalized atmospheric Rosenbluth method (GARM)

GARM (see [131]) is a generalized implementation of PERM in section 9, using generalized atmospheres. A particular implementation can use either plaquette atmospheres (see figure 13), or generalized atmospheres such as in figures 14 and 15. If endpoint atmospheres are used, then GARM reduces to PERM. The basic operation of GARM is predicated on a generalized counting formula which is derived below.

Consider a self-avoiding walk s and its associated generalized atmospheric statistics $a_+^g(s)$ and $a_-^g(s)$ as illustrated in figures 14 and 15. A new walk s' can be constructed from s by inserting a positive atmospheric edge into s, as illustrated in figure 26. In this case s' is a *successor* of s, and $|s'| = |s| + 1$. We call s a *predecessor* of s'. This construction will be called a *positive atmospheric move* (in the generalized atmosphere).

If S denotes the state space of walks from the origin, then positive atmospheric moves define a graph G_g on S with two vertices (walks) connected by an edge if the first is a predecessor of the second (see figure 22). We call this association between predecessors and successors *linkages* in the *atmospheric graph* on the state space S.

Negative atmospheric moves are similarly defined, but by deleting a negative atmospheric edge in the walk instead. Since the last edge in a walk is always a negative atmospheric edge, it is always possible to perform a negative atmospheric move, unless the walk has length 0 and is only the vertex at the origin. By recursively performing negative atmospheric moves, a walk is reduced until it has length 0. Since each negative atmospheric move can be reversed to a positive atmospheric move, this observation implies that the generalized positive atmospheric move is irreducible; there is a path in the atmospheric graph G_g from the origin to any given walk.

Consider the following sampling scheme in S. Let ϕ_0 be the walk of length 0 and also be the *initial state*. Generate states (walks) $\phi_1, \phi_2, \ldots$ by applying positive atmospheric moves: ϕ_j is obtained from ϕ_{j-1} by a uniformly chosen positive atmospheric move on ϕ_{j-1}.

This generates a sequence $\phi = \phi_0\phi_1 \cdots \phi_N$ of length $N+1$ in S where the state ϕ_j is a walk of length j. Since the state ϕ_j is obtained from ϕ_{j-1} by uniformly selecting a randomly chosen positive atmospheric move, the conditional probability that ϕ_j follows ϕ_{j-1} is

$$P(\phi_j|\phi_{j-1}) = \frac{1}{a_+^g(\phi_{j-1})}. \tag{196}$$

The probability to sample a given sequence ϕ is therefore

$$\Pr(\phi) = \prod_{j=1}^{|\phi|-1} P(\phi_j|\phi_{j-1}) = \prod_{j=1}^{|\phi|-1} \left[\frac{1}{a_+^g(\phi_{j-1})}\right], \tag{197}$$

where $|\phi| = N+1$ is the *number* of states in the sequence ϕ.

The final state of the sequence ϕ is a specific state $\phi_N \equiv \tau$, and the probability that a sequence terminates in the state τ is given by

$$\Pr(\tau) = \sum_{\phi\to\tau} \prod_{j=1}^{|\phi|-1} \left[\frac{1}{a_+^g(\phi_{j-1})}\right], \tag{198}$$

where the sum is over all sequences ϕ in S which starts as the trivial state ϕ_0 of one vertex in the origin and final state which is τ.

The key to the algorithm is to assign a weight $W(\phi)$ to each sequence: in the case here, we shall show that it is appropriate to define the weight of a given sequence ϕ by

$$W(\phi) = \prod_{j=1}^{|\phi|-1} \left[\frac{a_+^g(\phi_{j-1})}{a_-^g(\phi_j)}\right]. \tag{199}$$

Each factor in this product is the ratio of the positive atmosphere of the current state, divided by the negative atmosphere of the next state. This choice of the weight leads to the following lemma:

Lemma 10.1. *The average $\langle W(\phi)\rangle$ of the weight of sequences that end in the state τ is unity:*

$$\langle W(\phi)\rangle = \sum_{\phi\to\tau} W(\phi)Pr(\phi) = 1.$$

Proof. Consider the average

$$\langle W(\phi)\rangle = \sum_{\phi\to\tau} \prod_{j=1}^{|\phi|-1} \left[\frac{a_+^g(\phi_{j-1})}{a_-^g(\phi_j)}\right]\left[\frac{1}{a_+^g(\phi_{j-1})}\right] = \sum_{\phi\to\tau} \prod_{j=1}^{|\phi|-1} \left[\frac{1}{a_-^g(\phi_j)}\right]. \tag{200}$$

The last term can now be interpreted as the probability of the walk τ that is reduced to the single vertex ϕ_0 by uniformly applying negative atmospheric moves. This probability is 1,

Table 2. Approximate enumeration with GARM.

n	c_n	$\langle W\rangle_n$
0	2	1
1	4	4
2	12	12
3	36	36.0023
4	100	100.005
5	284	283.969
6	780	780.187
7	2172	2173.69
8	5916	5918.86
9	16 268	16 275.1
10	44 100	44 098.6
11	120 292	$1.203\,44 \times 10^5$
12	324 932	$3.250\,39 \times 10^5$
13	881 500	$8.816\,24 \times 10^5$
14	2374 444	$2.374\,73 \times 10^6$
15	6416 596	$6.416\,77 \times 10^6$
16	17 245 332	$1.724\,57 \times 10^7$
17	46 466 676	$4.645\,54 \times 10^7$
18	124 658 732	$1.246\,46 \times 10^8$
19	335 116 620	$3.350\,45 \times 10^8$
20	897 697 164	$8.975\,97 \times 10^8$
21	2408 806 028	$2.408\,89 \times 10^9$
22	6444 560 484	$6.442\,50 \times 10^9$
23	17 266 613 812	$1.726\,76 \times 10^{10}$
24	46 146 397 316	$4.616\,92 \times 10^{10}$
25	123 481 354 908	$1.236\,61 \times 10^{11}$

since the positive atmospheric moves are irreducible, and since every sequence of negative atmospheric moves starting in τ necessarily terminates in the trivial walk. □

By summing $\langle W(\phi)\rangle$ over all last states τ which are walks of length n, one obtains the counting formula

$$\sum_{|\tau|=n} \langle W(\phi)\rangle = \sum_{\substack{\phi\to\tau \\ |\tau|=n}} \prod_{j=1}^{|\phi|-1} \left[\frac{1}{a_-^g(\phi_j)}\right] = c_n, \tag{201}$$

where c_n is the number of walks of length n. Thus, by determining the average of the weights in an implementation of this algorithm, the numbers c_n are estimated. In table 2, the results of a simulation using GARM are compared to exact enumeration data from [83]. In this case, GARM sampled one million sequences of length $n = 25$ to compute average weights as in equation (201).

If the entire argument above is repeated for endpoint atmospheres, then the Rosenbluth algorithm is recovered. It is not clear that the plaquette atmospheres can be used; the positive and negative plaquette atmospheric moves are not irreducible on the set of walks since they leave the endpoints of the walk fixed in the lattice.

We call the implementation using generalized atmospheric moves GARM, an acronym for 'generalized atmospheric Rosenbluth method'.

The implementation of GARM is now similar to the implementation of the Rosenbluth algorithm.

Algorithm 10.2 (GARM). This implementation of GARM starts N walks of length at most n.

(1) Put $W = 1$ and let ϕ_0 be the trivial walk which is the vertex at the origin, and determine ϕ_1 by uniformly selecting a positive atmospheric move and executing it. Update W by multiplying it by $\left[a_+^g(\phi_0)/a_-^g(\phi_1)\right]$.
(2) Determine the positive atmosphere of ϕ_{j-1} and suppose it has size $a_+^g(\phi_{j-1})$. Select a positive atmospheric move uniformly and construct ϕ_j by executing it. Determine the size of the negative atmosphere of ϕ_j, $a_-^g(\phi_j)$.
(3) If $a_+^g(\phi_{j-1}) = 0$ then the state is trapped and the sequence ϕ is terminated. A new sequence is started at step (1).
(4) Put $W = W\left[a_+^g(\phi_{j-1})/a_-^g(\phi_j)\right]$.
(5) If $j < n$, then determine the next state by going to step (2). If $j = n$ then a walk of length n and weight W has been generated. Continue by starting a new walk at step (1) until N walks are generated.

The implementation in algorithm 10.2 allows for the possibility of trapped conformations with zero positive atmospheres; this rarely occurs in the case of generalized atmospheres. This means that trapped conformations are encountered only rarely and attrition is not a serious problem. Since the calculation of generalized atmospheres for walks of length n is $O(n)$, it follows that the computational effort in generating a single sequence with final walk of length n is $O(n^2)$. GARM slows down significantly with increasing length of the sampled walks. Implementations using the techniques developed by Clisby [18] should improve on this.

10.1. Flat histogram generalized atmospheric Rosenbluth method (flatGARM)

Variances of the computed weights $\langle W\rangle$ in a GARM simulation do not increase as quickly as in the Rosenbluth algorithm. However, with increasingly long sequences, variances do eventually increase, and these make good estimates of average weights more difficult. As in the case of PERM, enrichment and pruning can be introduced to improve the sampling, giving a flat histogram method similar to flatPERM. This is the flatGARM algorithm [131].

GARM with pruning and enrichment proceeds as for PERM, by tracking the weights of a given sequence. If this weight grows too small, then the sequence can be pruned, and if the weight grows too large, then it can be enriched.

Suppose that the jth state ϕ_j in a sequence has weight $W(\phi_j)$. Then the implementation proceeds by the calculation of the parameter r

$$r = \frac{W(\phi_j)}{\langle W\rangle_j}, \tag{202}$$

where $\langle W\rangle_j$ is the average of the weights of the jth state in all sequences generated thus far (including the current sequence). That is, the average is given by

$$\langle W\rangle_j = \frac{1}{N}\sum_{k=1}^{N} W\left(\phi_j^k\right), \tag{203}$$

where ϕ_j^k is the jth state in the kth sequence ϕ^k.[2]

[2] Observe that even if the number of sequences was increased by enrichment that all weights are summed up, but that they are divided by the number of *started* sequences, including the number of pruned sequences.

Calculating $W(\phi_j)$ to determine $\langle W\rangle_j$ requires the negative atmosphere of the state ϕ_{j+1}, which have not been constructed yet—to avoid this problem, the algorithm may be implemented by constructing the next state ϕ_{j+1}, then to determine enrichment at state ϕ_j (and then discarding ϕ_{j+1}).

Alternatively, extrapolating the negative atmosphere of the not yet constructed ϕ_{j+1} from ϕ_j has also been shown to work well in applications. For generalized atmospheres it is not unreasonable to suppose that $a_+^g(\phi_{j+1}) = a_-^g(\phi_j)+1$, and this guesstimate works well in actual simulations (it has the advantage that we do not need to construct ϕ_{j+1} before deciding on pruning or enrichment).

Once the parameter r is computed, pruning and enrichment are implemented as follows: if $r < 1$, then retain the current sequence with probability r and update the weight $W(\phi_j) \to W(\phi_j)/r$. Otherwise, prune the sequence (with probability $1-r$).

If $r > 1$ then enrich the sequence: compute $c = \lceil r\rceil$ with probability $r - \lfloor r\rfloor$, and $c = \lfloor r\rfloor$ otherwise. Make c copies of the sequence, each with weight $W(\phi_j)/c$ and continue growing from each, recursively pruning and enriching at each step.

The implementation of flatGARM proceeds as follows:

Algorithm 10.3 (flatGARM). Determine the number of sequences to be started, and n, the maximum length of each sequence.

(1) Suppose that $N-1$ sequences have been started, and that the Nth sequence will be grown from the state ϕ_0 which is the trivial walk composed of one vertex at the origin.

(2) Suppose that the current sequence has been grown to the jth state ϕ_j of length $j < n$, and weight $W(\phi_j) = W_j$.

(3) Compute $r = W_j/\langle W\rangle_j$, where $\langle W\rangle_j$ is the average weight of all states of length j encountered thus far, including the current state ϕ_j.

(4) Suppose that $r > 1$: put

$$c = \begin{cases} \lceil r\rceil, & \text{with probability } r - \lfloor r\rfloor, \\ \lfloor r\rfloor, & \text{otherwise.} \end{cases}$$

Make c copies of ϕ_j each of weight W_j/c. Grow each copy of ϕ_j independently using GARM to the next states ϕ_{j+1} and enrich and prune each from step (2) above. If the sequence has reached its desired length, then start a new sequence at step (1).

(5) If $r \leqslant 1$ then prune: with probability $1-r$ prune the current sequence and start growing the next sequence from step (1) above. With probability r put $W_j \to W_j/r$ and continue growing the sequence using GARM to the next state ϕ_{j+1}. If the sequence has reached its desired length, then start a new sequence at step (1). Otherwise, proceed by enrichment and pruning from step (2).

This implementation of flatGARM introduces some attrition of started sequences due to pruning, but the parameter r is designed such that it produces a flat histogram over the lengths of the walks: the pruned sequences are eventually replaced by enriched sequences.

A major advantage over PERM and flatPERM encountered here is that correlations are suppressed in the enrichment process. While endpoint atmospheric moves in PERM or flatPERM leave initial parts of the walks unchanged up to the enrichment point, producing persistent correlations between these walks, the generalized atmospheric moves quickly change these parts of the walk, so that correlations between enriched copies soon become statistically of lesser significance.

A thermal implementation of GARM is similarly obtained by computing estimates of the partition function at a fugacity β:

$$Z_n^{\text{est}}(\beta) = \frac{1}{N}\sum_{k=1}^{N} W(\phi_j^k)e^{\beta E(\phi_j^k)} = \langle We^{\beta E}\rangle_j, \tag{204}$$

where ϕ_j^k is the jth state of length n in the kth sequence ϕ^k. In algorithm 10.3, the calculation of r is replaced by

$$r = \frac{W(\phi_j)e^{\beta E(\phi_j)}}{Z_n^{\text{est}}(\beta)} \tag{205}$$

for state ϕ_j. The rest of the implementation is unchanged. This will enrich and prune states such that a flat histogram over the partition function at fugacity β is obtained.

10.2. Introducing neutral atmospheres into GARM and flatGARM

The discussion so far has centred on the implementation of sampling along sequences ϕ using positive atmospheres, and computing weights by using negative atmospheres. This can be generalized by the introduction of a neutral atmosphere in the algorithm.

Let a_0^g be an arbitrary neutral atmosphere defined on walks, for example the pivot neutral atmosphere, or the generalized exchange neutral atmosphere. GARM or flatGARM can be implemented by choosing at each step either a positive atmospheric move or a neutral atmospheric move with probabilities

$$P_+ = P(\text{positive atmospheric move}) = \frac{a_+^g(\phi_j)}{a_+^g(\phi_j) + a_0^g(\phi_j)}, \tag{206}$$

$$P_0 = P(\text{neutral atmospheric move}) = \frac{a_0^g(\phi_j)}{a_+^g(\phi_j) + a_0^g(\phi_j)}. \tag{207}$$

The corresponding weight of a sequence ϕ is given by

$$W(\phi) = \prod_{j=1}^{|\phi|-1} \frac{a_+^g(\phi_{j-1}) + a_0^g(\phi_{j-1})}{a_-^g(\phi_j) + a_0^g(\phi_j)}. \tag{208}$$

The mean weight in this implementation, $\langle W(\phi)\rangle$ over all sequences ending in a state of length n, is again equal to c_n, and the proof of this is similar to that of lemma 10.1. Further generalizations of this give the GAS-algorithm, which is discussed below. A flat-histogram implementation with neutral atmospheric moves follows the steps in algorithm 10.3, using enrichment and pruning to sample asymptotically uniform from walks of given length. Since the implementation of generalized positive atmospheres is irreducible, the addition of neutral atmospheric moves does not change this, but level j, denoted by ϕ_j, in a realized sequence ϕ, is not a walk of length j anymore, but will generally have length less than j. This implies that the introduction of neutral atmospheric moves will improve sampling at smaller lengths, and that longer sequences ϕ will be needed to collect sufficient data at large values of n. One may introduce a parameter to bias the sampling for or against neutral atmospheric moves by suitably adjusting the expressions for P_+ and P_0 above.

10.3. The microcanonical implementation of flatGARM

The flatGARM algorithm can be implemented to collect data in the microcanonical ensemble. Let $c_n(m) \equiv c_{n,m}$ be the number of walk of length n and 'energy' $E = m$. A microcanonical

implementation of flatGARM will proceed as above, but with c_n replaced by $c_{n,m}$: the energy E of the sequence ϕ is tracked such that state ϕ_j has energy $E(\phi_j)$.

An estimator for $c_{n,m}$ is obtained from equation (201) by noting that by restricting the sum to sequences ending in state τ of energy $m = E(\tau)$, one obtains

$$\langle W(\phi)\rangle = \sum_{\phi\to\tau} W(\phi)Pr(\phi) = \sum_{\phi\to\tau}\prod_{j=1}^{|\phi|-1}\left[\frac{1}{a_-^g(\phi_j)}\right] = 1, \tag{209}$$

where $Pr(\phi)$ is given by equation (198).

The proof now follows the same arguments as in lemma 10.1. Summing the left-hand side above over all sequences ϕ ending in the state τ of length n and energy m then shows that

$$\sum_{\substack{\phi\to\tau\\ |\tau|=n\\ E(\tau)=m}} \langle W(\phi)\rangle = \langle W(\phi)\rangle_{n,m} = c_{n,m} \tag{210}$$

since there are $c_{n,m}$ states τ of length n and energy m, and where $\langle W(\phi)\rangle_{n,m}$ is the mean weight of sequences ending in states of length n and energy m.

Thus, the estimator for the number of walks of length n and energy m is given by the average weight

$$c_{n,m}^{\text{est}} = \frac{1}{N}\sum_{k=1}^{N} W_{n,m}(\phi^k), \tag{211}$$

where ϕ^k is the kth sequence, and $W_{n,m}(\phi^k)$ is the weight of the sequence ϕ^k when ϕ^k is a state of length n and energy m.

To generate a flat histogram in both energy and length, suppose that the current state in a simulation is ϕ_j of length j, and compute the parameter r by putting

$$r = \frac{W(\phi_j)}{\langle W_{j,m}\rangle}, \tag{212}$$

where $W(\phi_j)$ is the weight of state ϕ_j of length j and energy m and $\langle W_{j,m}\rangle$ is the average weight of states of length j and energy m, measured so far in this simulation, including the weight of ϕ_j. If $r < 1$, then the sequence can be pruned with probability r as before, and if $r > 1$ it can be enriched. Thus, walks of energy m are enriched in the ensemble if they have large weight and pruned if they have small weight. The enrichment and pruning conspires to keep r of order unity, and a sequence will have weight close to the average weight.

The implementation of the algorithm proceeds now as follows:

Algorithm 10.4 (Microcanonical flatGARM).

(1) Implement GARM and suppose that $N-1$ sequences have been started and completed and that the current sequence ϕ has current state a walk ϕ_j of length j, energy m and weight $W_j \equiv W(\phi_j)$.
(2) Compute $r = W_j/\langle W_{j,m}\rangle$, the average weight of all states of length j and energy m encountered thus far, including the current state ϕ_j.
(3) Suppose that $r > 1$: put

$$c = \begin{cases} \lceil r\rceil, & \text{with probability } r - \lfloor r\rfloor,\\ \lfloor r\rfloor, & \text{otherwise.}\end{cases}$$

Make c copies of ϕ_j each of weight W_j/c. Grow each copy of ϕ_j independently using GARM to the next states ϕ_{j+1} and enrich and prune each from step (1) above.

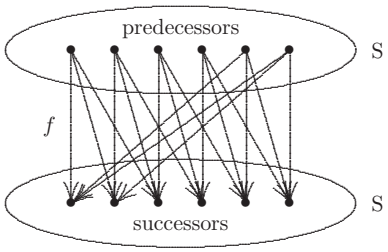

Figure 27. Generalized atmospheric moves in the GAS algorithm sets up a map $f : S \to S$ which maps predecessors in S to successors in S. f may be represented as a digraph as illustrated above. Observe that the outdegree of any predecessor vertex in S is equal to the indegree of any successor vertex. This is necessarily true, since every atmospheric move is reversible. That is, if $s \in S$ is a state, then indeg s = outdeg s, since every arc into s is also an arc out of s.

(4) If $r \leqslant 1$ then prune: put $W_j \to W_j/r$ and determine the next state ϕ_{j+1} with probability r by using GARM. Then proceed from step (1) above. Otherwise, prune the current sequence with probability $1 - r$ and start growing the next sequence by using GARM from step (1) above.

An implementation of this algorithm with endpoint atmospheres gives a microcanonical implementation of flatPERM. The implementation with other atmospheric moves requires careful coding to optimize the efficiency of the algorithm, since the calculation of generalized atmospheres are more demanding of CPU time.

11. Generalized atmospheric sampling

Generalized atmospheric sampling (GAS) is an implementation of general atmospheric moves to sample walks (states) along a sequence ϕ. The method generalizes GARM by implementing negative atmospheric moves in addition to neutral and positive atmospheric moves.

Consider a self-avoiding walk s together with its associated atmospheric statistics $a_+^g(s)$, $a_0^g(s)$ and $a_-^g(s)$ of positive, neutral and negative atmospheres together with their corresponding atmospheric moves. We require that (1) the set of atmospheric moves is irreducible, (2) that every positive atmospheric move is reversible by a corresponding negative atmospheric move, and vice versa, and (3) that every neutral atmospheric move is reversible by a corresponding neutral atmospheric move.

If s is a given walk, then a walk s' can be constructed from s by selecting a positive, neutral or negative atmospheric move. Generally, the atmospheric move may insert or delete edges in s, or in the case of a neutral atmosphere, change the conformation of s in some way which preserves its length. We call s the *predecessor* of s', and s' is the *successor* of s.

Since each atmospheric move is reversible, s is both a predecessor and a successor of s' (and vice versa).

Let S be the state space of all walks, then the atmospheric moves define linkages in S as in figure 22. We now generalise the graph of linkages. Let $f : S \to S$ be a map which maps the states in S to S such that $(s, s') \in f$ if s is a predecessor of s'. Then f may be represented as a digraph with vertex set two copies of S and arcs from vertices in a copy S (the predecessor vertices) to vertices in the second copy of S (the successor vertices). We illustrate this in figure 27.

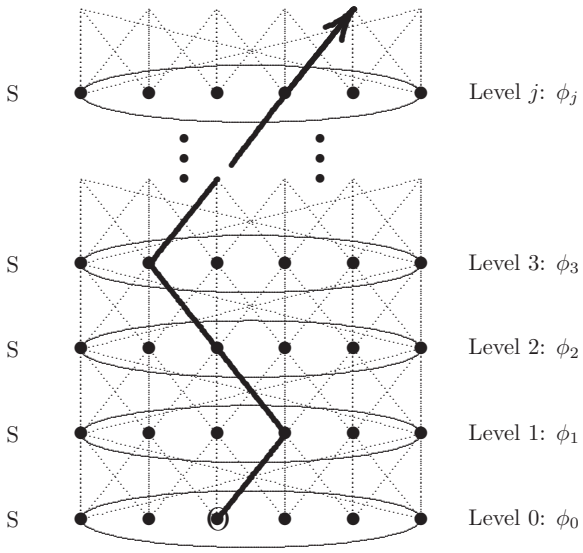

Figure 28. Sampling in the GAS-algorithm. The algorithm is initiated in state ϕ_0 in level 0, marked by O. A successor state in level 1 is selected as the next state by applying an atmospheric move (positive, neutral or negative). If the current level is j, then the state ϕ_{j+1} in level $j = 1$ is selected from the successors of state ϕ_j. Eventually the sequence ϕ of states is a path in the diagram as illustrated above. The dotted arcs are possible atmospheric moves and are directed from bottom to top. The sequence ϕ is denoted by the solid path, and steps up through the levels. Observe that the indegree of each vertex in this digraph is equal to the outdegree. If $s \in S$ is a state in level j, then indeg s = outdeg $s = a_+^g(s) + a_0^g(s) + a_-^g(s)$.

Let $\phi_0 \in S$ be an initial state in the algorithm. Successors of ϕ_0 are states ϕ_1 which can be reached from ϕ_0 by implementing a (positive, neutral or negative) atmospheric move. Once ϕ_1 has been selected as the next state, then ϕ_2 can be selected from amongst the successors of ϕ_1. Generally, ϕ_{j+1} is selected from amongst the successors of ϕ_j, but necessarily with uniform probability. This process builds a sequence $\phi = \phi_0\phi_1\phi_2\cdots\phi_j\cdots$ of states by repeated compositions of the map f defined in figure 27 and above.

The *level* of a state ϕ is its position in the sequence $\phi = \phi_0\phi_1\phi_2\cdots\phi_j\cdots$. For example, ϕ_0 has level 0 and ϕ_j has level j.

The sampling of states from the successors of the current state is the basic operation of the GAS-algorithm. When the jth state is sampled, the algorithm is said to *sample in level* j, and we illustrate this in figure 28: the algorithm starts in level zero, and then realizes a sequence $\phi = \phi_0\phi_1\phi_2\cdots\phi_j\cdots$ by sampling successors level by level; the state ϕ_j is sampled from the jth level. Finally, the sequence ϕ is a directed path through the levels in the digraph in figure 28 such that state ϕ_j is in level j.

Observe that at this point S is the collection of all walks of arbitrary length, an infinite set. A small modification to the algorithm can be made so that S is a finite set: define all walks in S of length n to have zero positive atmospheres. That is, if the state ϕ_j of length n is sampled by GAS, then its positive atmosphere is zero. In this case, the number of walks which can be reached from the trivial walk of length zero has length at most n, and this collection of walks is finite. This modification does not change the GAS sampling, as long the atmospheric moves are still collectively irreducible on the set of walk of length at most n. The sampling is still

along a sequence ϕ from level to level as in figure 28, but now S is a finite set. We will use these observations to define a flat histogram version of GAS below.

Suppose that state ϕ_j in level j in GAS-sampling has been realized. Introduce the parameter β (possibly dependent on the number of edges of state ϕ_j) and perform an atmospheric move with probabilities

$$P_+ = P(\text{positive atmospheric move}) = \frac{\beta a_+^g(\phi_j)}{a_-^g(\phi_j) + a_0^g(\phi_j) + \beta a_+^g(\phi_j)}, \tag{213}$$

$$P_0 = P(\text{neutral atmospheric move}) = \frac{a_0^g(\phi_j)}{a_-^g(\phi_j) + a_0^g(\phi_j) + \beta a_+^g(\phi_j)}, \tag{214}$$

$$P_- = P(\text{negative atmospheric move}) = \frac{a_-^g(\phi_j)}{a_-^g(\phi_j) + a_0^g(\phi_j) + \beta a_+^g(\phi_j)}, \tag{215}$$

which are normalized to sum up to unity.

The purpose of the GAS-algorithm is to compute a weight $W(\phi)$ for a realized sequence ϕ of states. Implementation of the algorithm is as follows:

Algorithm 11.1 (GAS). This algorithm samples a long sequence $\phi = \phi_0\phi_1\phi_2\cdots\phi_j\cdots$ in the state space S of walks where state ϕ_j is said to be in level j.

(1) Define the state ϕ_0 in level 0 (normally the trivial walk composed of the single vertex at the origin with length 0 edges). Set β at a convenient value, and let L be the desired length (number of states) in the sequence ϕ.
(2) Initialize the *weight* W of the sequence ϕ by putting $W_0 = 1$.
(3) If state ϕ_j in level j and of weight W_j has been determined, then compute the atmospheres $a_+^g(\phi_j)$, $a_0^g(\phi_j)$ and $a_-^g(\phi_j)$.
(4) Update W_j by putting

$$W'_{j+1} = \left(a_-^g(\phi_j) + a_0^g(\phi_j) + \beta a_+^g(\phi_j)\right) W_j.$$

(5) Compute the probabilities in equations (213)–(215). Use these to determine whether the next atmospheric move is positive, neutral or negative. Perform an atmospheric move of the kind selected by uniformly choosing a move from list of possible moves. This gives the state ϕ_{j+1}.
(6) Define the function σ in the sequence ϕ by

$$\sigma(\phi_j, \phi_{j+1}) = \begin{cases} -1, & \text{if} \quad \phi_{j+1} \text{ follows } \phi_j \text{ through } a_+, \\ +1, & \text{if} \quad \phi_{j+1} \text{ follows } \phi_j \text{ through } a_-. \end{cases}$$

That is, if $\phi_j \to \phi_{j+1}$ through a positive (negative) atmospheric move, then $\sigma(\phi_j, \phi_{j+1}) = -1(+1)$. Update the weight by

$$W_{j+1} = \frac{W'_{j+1}\beta^{\sigma(\phi_j,\phi_{j+1})}}{\left(a_-^g(\phi_{j+1}) + a_0^g(\phi_{j+1}) + \beta a_+^g(\phi_{j+1})\right)}.$$

This produces the next state ϕ_{j+1} in the sequence ϕ.
(7) If the sequence has reached a desired level, say $j = L$, then terminate the algorithm. It has generated a sequence ϕ of weight W_L. Otherwise, proceed to step (3) to find the next state.

Define ℓ_j to be length (number of edges) of state ϕ_j (which is a walk in S). Define $|\phi|$ to be the number of levels in a sequence realized by GAS. If GAS realizes a sequence ϕ with $|\phi|$ levels, then the weight of ϕ is

$$W(\phi) = \left[\prod_{j=0}^{|\phi|-1}\left[\frac{a_-^g(\phi_j) + a_0^g(\phi_j) + \beta a_+^g(\phi_j)}{a_-^g(\phi_{j+1}) + a_0^g(\phi_{j+1}) + \beta a_+^g(\phi_{j+1})}\right]\right]\prod_{j=0}^{|\phi|-1}\beta^{\sigma(\phi_j,\phi_{j+1})}. \quad (216)$$

Define $P(\phi)$ to be the number of positive atmospheric moves in ϕ, and $N(\phi)$ to be the number of negative atmospheric moves in ϕ. Then $P(\phi) - N(\phi)$ is equal to ℓ_L, the length of the final state ϕ_L in ϕ. The products above telescope down to the much simplified expression:

$$W(\phi) = \left[\frac{a_-^g(\phi_0) + a_0^g(\phi_0) + \beta a_+^g(\phi_0)}{a_-^g(\phi_L) + a_0^g(\phi_L) + \beta a_+^g(\phi_L)}\right]\beta^{N(\phi)-P(\phi)}. \quad (217)$$

Next, the probability of realizing a particular sequence ϕ is given by

$$P(\phi) = \left[\prod_{j=0}^{|\phi|-1}\left[\frac{1}{a_-^g(\phi_j) + a_0^g(\phi_j) + \beta a_+^g(\phi_j)}\right]\right]\beta^{P(\phi)} \quad (218)$$

since the probability of a positive atmospheric move is given by $\beta\big/\big(a_-^g(\phi_j) + a_0^g(\phi_j) + \beta a_+^g(\phi_j)\big)$ and the probability of a negative or neutral atmospheric move is given by $1\big/\big(a_-^g(\phi_j) + a_0^g(\phi_j) + \beta a_+^g(\phi_j)\big)$ at level j.

The expected value of the weight over all sequences terminating in the state τ is then given by

$$\begin{aligned}\langle W(\phi)\rangle_\tau &= \sum_{\phi:\phi_0\to\tau} W(\phi)P(\phi)\\ &= \sum_{\phi:\phi_0\to\tau}\left[\prod_{j=0}^{|\phi|-1}\left[\frac{1}{a_-^g(\phi_{j+1}) + a_0^g(\phi_{j+1}) + \beta a_+^g(\phi_{j+1})}\right]\right]\beta^{N(\phi)}, \quad (219)\end{aligned}$$

where the summation over $\phi : \phi_0 \to \tau$ is over all sequences starting from the trivial state ϕ_0 and ending in state τ.

Consider a sequence $\phi : \phi_0 \to \tau$ and reverse each step along it to get the backwards sequence $\phi' : \tau \to \phi_0$. Since the indegrees of any state in the derivative graph equal the outdegree of the same state, the atmospheres of any state in the sequence ϕ' are equal to the corresponding state in the forward sequence ϕ. Now each positive atmospheric step in ϕ is a negative atmospheric step in ϕ', and vice versa. Hence $N(\phi) = P(\phi')$ and $P(\phi) = N(\phi')$ and we can write the last summation as a sum over the backwards sequences ϕ':

$$\langle W(\phi)\rangle_\tau = \sum_{\phi':\tau\to\phi_0}\left[\prod_{j=0}^{|\phi'|-1}\left[\frac{1}{a_-^g(\phi'_{j+1}) + a_0^g(\phi'_{j+1}) + \beta a_+^g(\phi'_{j+1})}\right]\right]\beta^{P(\phi')}. \quad (220)$$

The summand in the above is the probability that a sequence ϕ' starting in the state τ will terminate in the trivial state ϕ_0 if positive atmospheric steps are given with probability

$$P^+ = \frac{\beta a_+^g(\phi'_{j+1})}{a_-^g(\phi'_{j+1}) + a_0^g(\phi'_{j+1}) + \beta a_+^g(\phi'_{j+1})}, \quad (221)$$

neutral atmospheric moves with probability

$$P^0 = \frac{a_0^g(\phi'_{j+1})}{a_-^g(\phi'_{j+1}) + a_0^g(\phi'_{j+1}) + \beta a_+^g(\phi'_{j+1})}, \quad (222)$$

and negative atmospheric moves with probability

$$P^- = \frac{a_-^g(\phi'_{j+1})}{a_-^g(\phi'_{j+1}) + a_0^g(\phi'_{j+1}) + \beta a_+^g(\phi'_{j+1})}. \tag{223}$$

If the set of atmospheric moves is irreducible, and if the mean probabilities of positive and negative atmospheric moves satisfy

$$\langle P^+\rangle_\ell \leqslant \langle P^-\rangle_\ell \tag{224}$$

over the entire range of lengths of walks in the chain, then this probability (that the chain terminates at the trivial state) is bigger than zero, since the atmospheric moves are reversible and the entire process is a (biased) random walk on the integers which is an ergodic Markov chain.

The above is in particular true if β in equation (223) satisfies

$$\beta \leqslant \frac{\langle a_1^g\rangle_\ell}{\langle a_+^g\rangle_\ell}\ \forall \ell, \qquad \text{so that} \quad \beta \leqslant \inf_\ell \left[\frac{\langle a_-^g\rangle_\ell}{\langle a_+^g\rangle_\ell}\right]. \tag{225}$$

In this event, the mean of the weight of the sequence ϕ ending in state τ is given by

$$\langle W(\phi)\rangle_\tau = P(\phi_0|\tau), \tag{226}$$

where $P(\phi_0|\tau)$ is the conditional probability that the (backwards) chain will terminate in state ϕ_0, given that it started in state τ. Summing this over all walks τ of length n shows that

$$\sum_{|\tau|=n} \langle W(\phi)\rangle_\tau = c_n \langle P(\phi_0|\tau)\rangle_n, \tag{227}$$

where c_n is the number of walks of length n, and where $\langle P(\phi_0|\tau)\rangle_n$ is the mean conditional probability that sequences starting in a state τ of length n will terminate in ϕ_0 stepping with backwards probabilities P^+, P^0 and P^-.

Similarly, for walks σ of length m it follows that

$$\sum_{|\sigma|=m} \langle W(\phi)\rangle_\sigma = c_m \langle P(\phi_0|\sigma)\rangle_m. \tag{228}$$

If the sequence ϕ is asymptotically long, then the mean conditional probabilities $\langle P(\phi_0|\tau)\rangle_n$ and $\langle P(\phi_0|\sigma)\rangle_m$ become independent of n and m, since the backwards chain is a recurrent Markov process if β is small enough. Thus, these factors cancel when the ratio of equations (227) and (228) is taken. This gives

$$\frac{c_n}{c_m} = \frac{\sum_{|\tau|=n}\langle W(\phi)\rangle_\tau}{\sum_{|\sigma|=m}\langle W(\phi)\rangle_\sigma}. \tag{229}$$

In particular, if $m = 0$, then $c_0 = 1$ and

$$c_n = \frac{\sum_{|\tau|=n}\langle W(\phi)\rangle_\tau}{N_0}, \tag{230}$$

since the weight of a sequence terminating in the trivial state ϕ_0 is one, and where N_0 is the number of visits of the sequence ϕ to the state ϕ_0. This last expression is simply the ratio of the accumulated weight of states of length n along the sequence ϕ, to the accumulated weight of the trivial state.

In practical implementations, the total (unnormalized) accumulated weight $\sum_{|\tau|=j}\langle W(\phi)\rangle_\tau$ is collected along sequences ϕ realized by the algorithm. Ratios of these accumulated weights produce estimates of ratios of c_n as in equations (229) and (230). Normally a simulation will proceed by generating a sequence ϕ with L levels (where L is

large), and accumulated weights are binned for each value of n. This gives the accumulated weights in equation (229) from which c_n can be estimated.

The implementation of GAS proceeds by the realizing N independent sequences ϕ and computing weights for each. This provides one with independent estimates for c_n/c_m, from which data analysis can be done. As for the GARM algorithm, GAS is in principle an approximate enumeration algorithm, and we shall see below that a flat histogram version can be implemented.

11.1. Flat histogram generalized atmospheric sampling (flatGAS)

In this section, a method for sampling walks by GAS of lengths $\ell \in [0, n_{\max}]$ is described. The algorithm will also produce a flat histogram over $(0, n_{\max})$, sampling states of length ℓ uniformly in $[0, n_{\max}]$, except at the endpoints of this interval.

Consider the interval $[0, n_{\max}]$, and suppose that GAS has been used to sample along a sequence ϕ so that state ϕ_j was sampled at level j. The length of ϕ_j is ℓ_j (this is the number of edges in the walk corresponding to ϕ_j).

A restriction on the lengths ℓ_j can be built into GAS as follows: *define* the positive atmospheres of states in S of length $\ell = n_{\max}$ to be equal to zero. In other words, if $\phi_j \in S$ and $\ell_j = n_{\max}$, then $a_+^g(\phi_j) = 0$.

This imposition is completely artificial in the sense that a (non-zero) positive atmosphere can be defined of states of length $n_{\max}$, but that we set this to be equal to zero. The effect of this is that states of lengths longer than $n_{\max}$ cannot be reached by GAS if it uses the trivial starting state ϕ_0 which is the trivial walk of length zero. Thus, with this change, GAS is not irreducible on walks anymore, and the algorithm will fail to generate sequences ϕ which include states of length longer than $n_{\max}$.

However, if we consider the state space $S(n_{\max})$ of walks of lengths in $[0, n_{\max}]$, then the algorithm is irreducible on $S(n_{\max})$ if any state ϕ_j of length $\ell_j \leqslant n_{\max}$ can be reduced to the trivial walk by the application of negative atmospheric moves. This turns out to be the case if either generalized atmospheric moves or endpoint atmospheric moves are used in GAS.

An implementation of GAS on $[0, n_{\max}]$ proceeds exactly as before, with the exception that states of length $n_{\max}$ have zero positive atmospheres. The algorithm operates by executing a (biased) random walk on the integers in $[0, n_{\max}]$, and because it is irreducible, it will sample all walks of lengths $\ell \in [0, n_{\max}]$ with positive probability. The aim is next to sample uniformly in $(0, n_{\max})$.

In GAS, the lengths of states are controlled by the parameter β defined in algorithm 11.1. This parameter is restricted by equations (224) and (225). We define the flatGAS algorithm by making β dependent on ℓ: in equation (224), choose β_ℓ such that

$$\langle P^+ \rangle_\ell = \langle P^- \rangle_\ell \tag{231}$$

and replace β by β_ℓ in equations (213)–(215). In particular, for each value of $\ell \in (0, n_{\max})$,

$$\beta_\ell = \frac{\langle a_-^g \rangle_\ell}{\langle a_+^g \rangle_\ell}, \tag{232}$$

while $\beta_0 = 1$ by default and where $\beta_{n_{\max}} = 0$. This implies that for $j = 1, 2, 3, \ldots, n_{\max} - 1$,

$$P_+ = P(\text{positive atmospheric move}) = \frac{\beta_{\ell_j} a_+^g(\phi_j)}{a_-^g(\phi_j) + a_0^g(\phi_j) + \beta_{\ell_j} a_+^g(\phi_j)}, \tag{233}$$

$$P_0 = P(\text{neutral atmospheric move}) = \frac{a_0^g(\phi_j)}{a_-^g(\phi_j) + a_0^g(\phi_j) + \beta_{\ell_j} a_+^g(\phi_j)}, \tag{234}$$

$$P_- = P(\text{negative atmospheric move}) = \frac{a_-^g(\phi_j)}{a_-^g(\phi_j) + a_0^g(\phi_j) + \beta_{\ell_j} a_+^g(\phi_j)}, \tag{235}$$

with the result that $\langle P_+ \rangle_\ell = \langle P_- \rangle_\ell$ for each $\ell \in (0, n_{\max})$.

With this choice of β_ℓ, for each value of $\ell = (0, n_{\max})$, GAS will select a longer walk as the next state with average probability equals to the average probability of selecting a shorter walk as the next state. Thus, GAS executes a random walk on $[0, n_{\max}]$ which is locally still biased, but on average is unbiased in the lengths of the states. Since the algorithm is ergodic on this interval, it will sample asymptotically from the uniform distribution on $(0, n_{\max})$ and visit the states of length 0 and $n_{\max}$ half as frequently as the states of lengths in $(0, n_{\max})$.

The implementation proceeds as follows:

Algorithm 11.2 (flatGAS). This algorithm samples along a sequence $\phi = \phi_0\phi_1\phi_2 \cdots \phi_j \cdots$ in the state space $S(n_{\max})$ of walks where state ϕ_j is said to be in level j and has length ℓ_j. Sequences are sampled until a total of N independent sequences have been completed.

(1) Define the state ϕ_0 in level 0 (normally the trivial walk composed of the single vertex at the origin with length 0 edges). Set β_ℓ at a convenient value for each $\ell \in [0, n_{\max}]$, and let L be the desired length (number of states) in the sequence ϕ.

(2) Initialize the *weight* W of the sequence ϕ by putting $W_0 = 1$.

(3) If state ϕ_j in level j and of weight W_j has been determined, then compute the atmospheres $a_+^g(\phi_j)$, $a_0^g(\phi_j)$ and $a_-^g(\phi_j)$. Note that $a_+^g(\phi_j) = 0$ of $\ell_j = n_{\max}$.

(4) Update W_j by putting

$$W'_{j+1} = \left(a_-^g(\phi_j) + a_0^g(\phi_j) + \beta_{\ell_j} a_+^g(\phi_j)\right) W_j.$$

(5) Compute the probabilities in equations (233)–(235). Use these to determine whether the next atmospheric move is positive, neutral or negative. Perform an atmospheric move of the kind selected by uniformly choosing a move from list of possible moves. This gives the state ϕ_{j+1}.

(6) Define the function σ in the sequence ϕ by

$$\sigma(\phi_j, \phi_{j+1}) = \begin{cases} -1, & \text{if} \quad \phi_{j+1} \text{ follows } \phi_j \text{ through } a_+, \\ +1, & \text{if} \quad \phi_{j+1} \text{ follows } \phi_j \text{ through } a_-. \end{cases}$$

That is, if $\phi_j \to \phi_{j+1}$ through a positive (negative) atmospheric move, then $\sigma(\phi_j, \phi_{j+1}) = -1(+1)$. Update the weight by

$$W_{j+1} = \frac{W'_{j+1} \beta_{\ell_{j+1}}^{\sigma(\phi_j, \phi_{j+1})}}{\left(a_-^g(\phi_{j+1}) + a_0^g(\phi_{j+1}) + \beta_{\ell_{j+1}} a_+^g(\phi_{j+1})\right)}.$$

This produces the next state ϕ_{j+1} in the sequence ϕ.

(7) If the sequence has reached a desired level, say $j = L$, then terminate the sequence, otherwise proceed to step (3) to find the next state. If the sequence was terminated, then update estimates for β_ℓ for each $\ell \in [0, n_{\max}]$ by computing

$$\beta_\ell = \frac{\langle a_-^g \rangle_\ell}{\langle a_+^g \rangle_\ell}.$$

With this new set of β_ℓ, start a new sequence from step (1). Repeat this until a desired number N of sequences have been realized. If each of the sequences ϕ is long, then each new sequence will generate a flatter histogram over the lengths $\ell \in [0, n_{\max}]$.

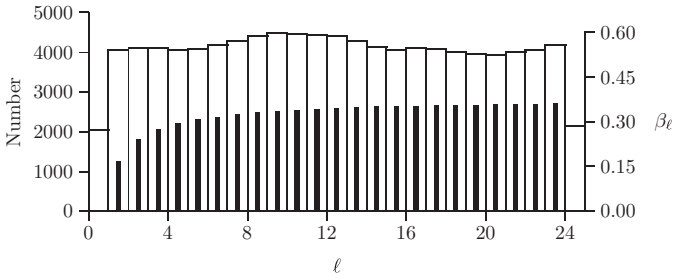

Figure 29. Flat histogram sampling in the flatGAS-algorithm. The solid bars give the values of β_ℓ as a function of ℓ on the right-hand side scale. $\beta_0 = 1$ by definition and is left away, since the probability of stepping from the walk of length 0 to a walk of length 1 is $1/2d$, independent of β_0. In this simulation, $n_{\max} = 24$ and so $\beta_{24} = 0$. The values of β_ℓ were computed by atmospheric ratios (see equation (232)) from an earlier run. The binning of states of length $\ell \in [0, 24]$ in a sequence of length 100 000 with β_ℓ as given are indicated by the open bars with scale on the left-hand axis. The states with $\ell = 0$ and $\ell = 24$ were sampled about half as frequently as those states with $\ell \in (0, 24)$.

Suppose that flatGAS has sampled along a sequence ϕ in $S(n_{\max})$ of length $|\phi|$, and with states ϕ_j of lengths ℓ_j at levels j. Then the weight of the sequence ϕ is given by

$$W(\phi) = \left[\frac{a_-^g(\phi_0) + a_0^g(\phi_0) + \beta_{\ell_0} a_+^g(\phi_0)}{a_-^g(\phi_L) + a_0^g(\phi_L) + \beta_{\ell_L} a_+^g(\phi_L)}\right] \prod_{j=0}^{|\phi|-1} \beta_{\ell_j}^{\sigma(\phi_j,\phi_{j+1})}. \tag{236}$$

The argument proceeds from here similarly as set out in GAS above: if the sequence ϕ is asymptotically long on the finite state space $S(n_{\max})$, then the mean conditional probability $\langle P(\phi_0|\tau)\rangle_n$ that the backwards sequence ϕ will terminate in ϕ_0 becomes independent of n, since the backwards chain is now an irreducible and recurrent Markov process in a finite state space. In particular, $\langle P(\phi_0|\tau)\rangle_n$ is asymptotically independent of ϕ_0, and if ratios of the mean accumulated weights are taken, then

$$\frac{\sum_{|\tau|=n}\langle W(\phi)\rangle_\tau}{\sum_{|\sigma|=m}\langle W(\phi)\rangle_\sigma} = \frac{c_n}{c_m}. \tag{237}$$

The implementation of flatGAS proceeds by realizing N independent sequences ϕ and computing weights for each while updating the values of β_ℓ following each sequence ϕ. If the initial choices for β_ℓ were erroneous, updated estimates quickly improve, and eventually a flat histogram of states on the interval $(0, n_{\max})$ is obtained, while the states of length 0 and $n_{\max}$ are visited about half as often on average (flatGAS performs a random walk on the integers in $[0, n_{\max}]$, with reflecting boundaries, and hence the states 0 and $n_{\max}$ will be visited half as often as those in $(0, n_{\max})$).

In figure 29, an example of flatGAS sampling of walks using generalized atmospheres is given. The length of the sequence was $L = 100\,000$ levels, and values of β_ℓ were computed from an earlier run using equation (232); these are denoted by the solid bars. The binning of states of length ℓ in figure 29 shows that flatGAS effectively performs a random walk on $\ell \in [0, n_{\max}]$, producing the flat histogram indicated by the open bars.

Estimates for c_n obtained by an endpoint atmospheric implementation of flatGAS is displayed in table 3. These data were taken from an implementation of flatGAS that realized 2000 sequences with $n_{\max} = 1000$, each sequence of length 5×10^8.

The flat histogram version of GAS is in every respect, such as GAS, GARM and flatGARM, an approximate enumeration method.

Table 3. Approximate enumeration with flatGAS.

n	c_n	Estimate
0	1	0.999 96
1	4	3.999 87
2	12	12.000 17
3	36	36.003 57
4	100	100.021
5	284	284.068
6	780	780.206
7	2172	2172.58
8	5916	5917.77
9	16 268	16 273.5
10	44 100	44 117.1
11	120 292	$1.203\,43 \times 10^5$
12	324 932	$3.250\,92 \times 10^5$
13	881 500	$8.819\,74 \times 10^5$
14	2374 444	$2.375\,69 \times 10^6$
15	6416 596	$6.419\,57 \times 10^6$
16	17 245 332	$1.725\,31 \times 10^7$
17	46 466 676	$4.648\,99 \times 10^7$
18	124 658 732	$1.247\,19 \times 10^8$
19	335 116 620	$3.352\,87 \times 10^8$
20	897 697 164	$8.981\,95 \times 10^8$
21	2408 806 028	$2.410\,12 \times 10^9$
22	6444 560 484	$6.448\,16 \times 10^9$
23	17 266 613 812	$1.727\,59 \times 10^{10}$
24	46 146 397 316	$4.616\,66 \times 10^{10}$
25	123 481 354 908	$1.235\,28 \times 10^{10}$

12. The pivot algorithm

The pivot algorithm [92, 102] is a dynamic Monte Carlo algorithm implementing neutral pivot atmospheric elementary moves (see figure 19) to sample fixed length walks along a Markov chain. The state space of the algorithm is the set of walks of given length n, with two walks considered equivalent if the first is a translation of the second in the hypercubic lattice.

The elementary moves of the pivot algorithm are called *pivot moves* and are implemented on a walk s by selecting a *pivot* v to cut s into two subwalks s_0 and s_1. Without loss of generality, one may suppose that s_1 is shorter than s_0. The shorter subwalk s_1 is subjected to a uniformly selected symmetry operation P from the point symmetry group of the hypercubic lattice (in three dimensions this is the octahedral group, and in d dimensions it has order $2^d d!$) to obtain the subwalk $P(s_1) = s_1'$. By translating s_1' to reconnect it with s_0 at v, the walk $s' = s_0 v P(s_1) = s_0 v s_1'$ is obtained as illustrated in figure 11.

If s' is a self-avoiding walk, then it is accepted with probability one as the next state in the Markov chain, otherwise, s is read again as the next state in the Markov chain. This rejection technique implies that the algorithm samples along an aperiodic Markov chain.

It is known that the neutral atmospheric pivot elementary moves of the pivot algorithm are irreducible on the set of walks of fixed length n (see [102]), and so the pivot algorithm is ergodic (irreducible and aperiodic).

Observe that the pivot algorithm is also reversible. If s' is obtained from s by the application of an elementary move, then by selecting the same pivot point in s' and reversing the pivot move, it will again produce s. The probability of obtaining s' from s in this case is $[2^{-d}/d!]/(n-1)$ since the pivot point can be selected from one of $(n-1)$ possible vertices, and the elementary move is selected from one of $2^d d!$ possible choices. Thus $P_{s\to s'} = P_{s'\to s} = [2^{-d}/d!]/(n-1)$.

On the other hand, if there is no pivot move which will produce s' if s is the current state, then $P_{s\to s'} = P_{s'\to s} = 0$.

In other words, for any two walks s and s' of length n, the transition probabilities of the algorithm are reversible, and

$$P_{s\to s'} = P_{s'\to s}. \tag{238}$$

This shows that the pivot algorithm samples along a symmetric Markov chain in the state space of self-avoiding walks of length n.

Since the pivot algorithm is both ergodic and symmetric, it follows from the fundamental theorem of Monte Carlo algorithms that the pivot algorithm samples asymptotically from the uniform distribution on the state space of walks of fixed length in the hypercubic lattice.

States sampled along a realization of a Markov chain by the pivot algorithm are correlated. Averages over the states in the Markov chain are asymptotically normally distributed about their means, and confidence intervals can be determined by calculating autocorrelation times and variances as set out in section 4.6.

The pivot algorithm is implemented as follows:

Algorithm 12.1 (The pivot algorithm).

(1) Initialize the algorithm with an arbitrary walk s_0 of length n.
(2) Choose a pivot v in s_n, and identify the shorter subwalk ν into which s_n is cut by v.
(3) Update ν by operating on it with a uniformly chosen element of the symmetry group of the hypercubic lattice. This operation creates a proposed next state s'.
(4) If s' is self-avoiding, then $s_{n+1} = s'$, otherwise $s_{n+1} = s_n$. Continue from step (2).

The pivot algorithm is an implementation of an elementary move defined by the pivot neutral atmospheres defined in section 3.2. If (s_n, s') is a linkage between s_n and s' defined by a pivot neutral atmosphere, then the algorithm operates by choosing s' uniformly and with rejection as the next state from the set of walks which have linkages with s_n.

It is known that the pivot algorithm converges faster than any other algorithm which uses local moves involving less than a fixed number of M edges [102]. The integrated autocorrelation time of the implementation in [102] is $\tau \propto O(n)$ in CPU-time units for walks of length n. An implementation of the algorithm by Kennedy [86] has an integrated autocorrelation time estimated to be $\tau \sim O(n^{0.57})$ in CPU-time units for walks of length n on the square lattice, and $\tau \sim O(n^{0.85})$ for walks on the simple cubic lattice.

An even faster implementation of the algorithm was produced in [18]. In this case, the basic pivot move can be performed in $O(\log n)$ CPU-time units, giving an integrated autocorrelation time of approximately $\tau \sim O(n^{0.19}\log n)$ CPU-time units on the square lattice and $\tau \sim O(n^{0.11}\log n)$ CPU-time units on the simple cubic lattice. These implementations were achieved in [18] by noting that the self-avoiding walk can be represented as a binary

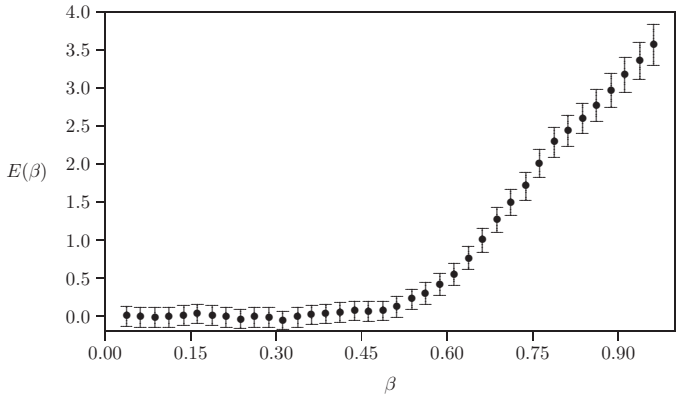

Figure 30. The mean energy per edge of an adsorbing walk of length $n = 120$ edges in the square lattice confined to the half-space $X \geqslant 0$. The mean energy is plotted against the fugacity $\beta = \log z$ and was computed by a multiple Markov chain implementation of the pivot algorithm for $\beta \in [0, 1]$. The implementation is of 10 chains equally spaced on the β-axis, and 125 000 000 attempted pivots were made on each chain with data collected every 250 attempts. The resulting time series of 500 000 points was analysed for each chain to compute the mean energies above. The data show a transition at $\beta_c \approx 0.55$ where the walk adsorbs on the $X = 0$ line. β is the fugacity in this implementation and $z = e^{\beta}$ is the activity of visits.

tree, where each walk is represented as the concatenation of two sub-walks of roughly equal length.

The pivot algorithm can be implemented via the Metropolis algorithm (algorithm 4.1) to sample interacting models of walks. If $Z_n(\beta) = \sum_v c_n(v)e^{\beta v}$ is the partition function of walks of length n at fugacity $\beta = \log z$, then the aim is to sample walks from the Boltzmann distribution in equation (120), where v_s is the energy of the walk s of length n. The implementation follows algorithm 4.1 and if the current state is s and an attempted pivot proposes t as the next state, then the probability that t is accepted as the next state is given by

$$P_{s\to t} = \min\{[P_\beta(t)/P_\beta(s)], 1\} = \min\{e^{\beta(v_t - v_s)}, 1\}, \tag{239}$$

where $P_\beta(s) = e^{\beta v_s}/Z_n(\beta)$. In other words, if $\beta < 0$, then transitions to proposed states with higher energies are more likely to be rejected.

The implementation of the pivot algorithm with umbrella sampling, or multiple Markov Chain sampling, on interacting models of walks follows the arguments in sections 4.3.3 and 4.3.4. In this incarnation, the algorithm is very useful in collecting data on critical phenomena in interacting models of walks, as it can be used to sample over intervals $[\beta_0, \beta_1]$ of the fugacity β; see for example [77, 145, 147]. In figure 30, a multiple Markov chain Monte Carlo implementation of the pivot algorithm was used to compute the mean energy of an adsorbing self-avoiding walk in the square lattice.

12.1. Cut-and-paste algorithms

A generalization of the pivot algorithm leads to cut-and-paste type algorithms for self-avoiding walks. The underlying idea is similar to pivots: walks in the canonical ensemble are sampled along a Markov chain using a dynamic Monte Carlo.

Cut-and-paste algorithms for self-avoiding walks are generalizations of the pivot algorithm. The general elementary move is as follows: cut a given walk of length n into N pieces or arbitrary length, rotate and reflect each piece independently by applying a symmetry

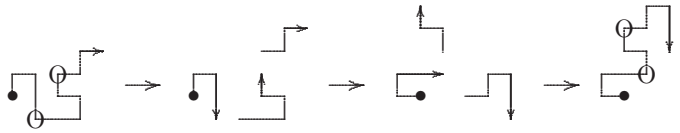

Figure 31. The elementary move of the cut-and-paste algorithm for self-avoiding walks. In this example, the walk is cut into three subwalks, which are reflected or rotated in the lattice, then shuffled, and then put back together into the next proposed walk.

operation from the point symmetry group of the lattice, and then attempt to reconnect the pieces in arbitrary order into a single self-avoiding walk. An example of the elementary move is given in figure 31. If a (non self-intersecting) walk is obtained, then it is the next state, otherwise the current walk is read again as the next state. This elementary move defines a cut-and-paste neutral atmospheric statistic on the walk.

13. The Beretti–Sokal algorithm

Grand canonical algorithms for self-avoiding walks implement length-changing atmospheric moves to sample from a distribution over the lengths of walks along a Markov chain in the state space of walks of arbitrary length. The simplest such algorithm implements the endpoint atmospheric moves in figure 12 as elementary moves, and is called the Berretti–Sokal algorithm [9].

The basic operation is as follows: let s be the current state, and attempt to append an edge in one of N directions with probability P_+, or delete the last edge of the walk, if it has length bigger than zero, with probability P_-, where $P_+ + P_- = 1$. A suitable choice for the probabilities is given by

$$P_+ = \frac{Ne^\beta}{1 + Ne^\beta} \qquad \text{and} \qquad P_- = \frac{1}{1 + Ne^\beta}. \tag{240}$$

The transition probabilities of the algorithm is given by

$$P_{s\to t} = \begin{cases} \dfrac{e^\beta \chi_t}{1 + Ne^\beta}, & \text{if an edge is added to } s, \\ \dfrac{1}{1 + Ne^\beta}, & \text{if an edge is removed from } s, \end{cases} \tag{241}$$

where $\chi_t = 1$ if t is a self-avoiding walk, and $\chi_t = 0$ otherwise.

Since the coordination number of the hypercubic lattice is $2d$, one choice is $N = 2d$ in P_+ and P_-, and whenever an edge is added, its direction is selected uniformly from the available $2d$ directions. A slight improvement is achieved if $N = (2d - 1)$ and backsteps are excluded from the choices for a positive atmospheric move. If a proposed positive move causes an intersection in a proposed next state, then the move is rejected and the current state is read again in the sequence of sampled states.

The algorithm may be implemented as follows:

Algorithm 13.1 (The Beretti–Sokal algorithm).

(1) Initialize the algorithm by choosing the first state s_0 to be the vertex at the origin. Then the length of s_0 is zero. Define the parameter β.
(2) Let s_n be the current state. Determine the probabilities P_+ and P_- in equation (240) and use them to determine the nature of the next move.

(3) If the next move is a negative endpoint atmospheric move, then delete the last edge in the algorithm to obtain the state s_{n+1}. Continue at step (2).
(4) If the next move is a positive endpoint atmospheric move, then choose one of the available lattice edges incident with the last vertex, and append it to the walk. If the result is a self-avoiding walk, then accept this as s_{n+1}, and continue from step (2). It the proposed state is not self-avoiding, then reject it, and put $s_{n+1} = s_n$, and then continue from step (2).

The algorithm samples along a Markov chain in the state space of walks of arbitrary length from the origin. The parameter β can be adjusted to control the average length of the walk in the simulation.

The probability of a positive endpoint atmospheric move is P_+, and the probability of a negative atmospheric move is P_-. In other words, if the walk t is obtained from s, and $|t| = |s| + 1$, then $P_{s\to t} = e^\beta P_{t\to s}$ since $P_{s\to t} = e^\beta/(1 + Ne^\beta)$ while $P_{t\to s} = 1/(1 + Ne^\beta)$. Hence it follows that $P_{s\to t} = e^{\beta(|t|-|s|)} P_{t\to s}$ or

$$e^{\beta|s|} P_{s\to t} = e^{\beta|t|} e^\beta P_{t\to s}. \tag{242}$$

This is the condition of detailed balance for the algorithm, and it proves that the algorithm samples along a Markov chain asymptotically from the Boltzmann distribution

$$P_\beta(s) = \frac{e^{\beta|s|}}{\sum_s e^{\beta|s|}} = \frac{e^{\beta|s|}}{\sum_n c_n e^{\beta n}}. \tag{243}$$

This distribution is normalizable only if $\sum_n c_n e^{\beta n} < \infty$, or in other words, if $-\infty < \beta < -\lim_{n\to\infty}[\log c_n]/n = -\log\mu$, where μ is the growth constant of self-avoiding walks defined in equation (2). In other words, the mean length of walks sampled by the Berretti-Sokal algorithm increases without bound if $\beta > -\log\mu$, in which case the sampling is not from a stationary distribution.

In practical applications there is a trade-off between sampling longer walks, and efficiency in the algorithm. If β is fixed close to $-\log\mu$, then the algorithm samples walks with longer average length, but along a time series with longer autocorrelations. Smaller (more negative) values of β reduce the autocorrelations, but produce samples of walks with shorter average length. In [9], it is argued that the autocorrelation times τ have order at least $\tau \gtrsim O(\langle n\rangle^2)$ in a simulation where the average length of the walks is $\langle n\rangle$. This was improved to $O(\langle n\rangle^2) \lesssim \tau \lesssim O(\langle n\rangle^{1+\gamma})$ in [139, 140].

Extensions of the Berretti–Sokal algorithm (by adding more than one edge at a time), and its application to interacting models of walks were given in [11].

The Berretti–Sokal algorithm can also be implemented using the Metropolis algorithm: append an edge to the last vertex of the current walk s_n. If the edge coincides with the last edge of the walk, then a negative atmospheric move is performed. Otherwise, the new edge extends the walk. If the new object is a self-avoiding walk t, then $s_{n+1} = t$ with probability e^β, otherwise $s_{n+1} = s_n$.

If the length of s_n is larger than zero, then the probability of negative atmospheric move is $1/2d$, and the probability of extending the walk s_n in a particular direction by a positive atmospheric move is $e^\beta/2d$. In other words, if the walk t can be obtained from s by a positive atmospheric move, then s can be obtained from t by a negative atmospheric move and $P_{s\to t} = [e^\beta/2d] = e^\beta P_{t\to s}$ so that the condition of detailed balance is given by

$$e^{\beta|s|} P_{s\to t} = e^{\beta|t|} P_{t\to s} \tag{244}$$

and it follows that for $\beta < -\log\mu$ the algorithm samples from a stationary Boltzmann distribution as above. This implementation is less efficient than the implementation above,

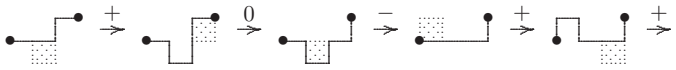

Figure 32. The elementary moves of the BFACF algorithm are induced by plaquette atmospheres. In this example, the marked plaquette incident with the walk induces a local move on the walk, either increasing or decreasing the length of the walk by two, or if the plaquette is a neutral atmospheric plaquette, leaving the length of the walk unchanged.

but can be easily adapted to sample interacting walks using umbrella sampling or multiple Markov chain sampling.

The mean length of walks sampled by the Berretti–Sokal algorithm depends on the value of β and is given by

$$\langle |s| \rangle_\beta = \frac{\sum_n n\, c_n e^{\beta n}}{\sum_n c_n e^{\beta n}}. \tag{245}$$

That is, the closer β is fixed to $-\log \mu$, the longer the walks. Sampling longer walks inevitably leads to a critical slowing down in the algorithm: the autocorrelation times become long and the measured data more uncertain. Thus β must be selected with some care; close enough to $-\log \mu$ to sample walks of sufficient length, but not too close to make data unreliable due to long autocorrelation times. A more generalized implementation of the Berretti–Sokal algorithm can be found in [113]: this implementation adds or removes up to Δn edges in each attempted iteration from the walk by determining generalized endpoint atmospheres.

14. The BFACF algorithm

The BFACF algorithm [4, 8] is a dynamic Monte Carlo algorithm which samples self-avoiding walks with fixed endpoints along a Markov chain by implementing the plaquette atmospheric moves in figure 13.

The plaquette atmospheres induce three kinds of elementary moves, namely one positive atmospheric move which increases the length of the walk by 2, a neutral atmospheric move, and a negative atmospheric move which decreases the length of the walk by 2. In figure 32, an example of BFACF-sampling is given.

The elementary moves of the BFACF algorithm do not move the endpoints of the walk, and these are fixed in the lattice. Generally, these elementary moves are not irreducible on the state space S_{0x} of walks with fixed endpoints 0 (the origin) and a lattice site x [101]. In the square lattice the moves are irreducible on S_{0x} for any fixed x [100]. In the cubic lattice the moves are irreducible whenever $\|x\|_\infty \geqslant 2$ [70].

Implementing the algorithm proceeds as follows: let s_n be the current walk of length $|s_n|$ edges. Choose with uniform probability an edge S in s_n, and enumerate the $2d-2$ atmospheric plaquettes incident with S. Some of the atmospheric plaquettes may be positive, some neutral and at most one may be negative. Choose, randomly, one of the $2d-2$ atmospheric plaquettes such that a (particular) positive atmospheric move is carried out with probability P_+, a negative atmospheric move is carried out with probability P_- and a (particular) neutral atmospheric move is carried out with probability P_0. This produces the state s'_n, and if it is a self-avoiding walk, then $s_{n+1} = s'_n$, otherwise $s_{n+1} = s_n$.

It remains to specify the probabilities P_+, P_- and P_0. Generally, the possible $2d-2$ atmospheric plaquettes have a probability attached, and the sum of these probabilities cannot exceed 1. The following cases can be enumerated:

- All $2d-2$ atmospheric plaquettes incident with S are positive. Thus $(2d-2)P_+ \leqslant 1$.
- One atmospheric plaquette incident with S is neutral; the remaining are all positive. This requires that $(2d-3)P_+ + P_0 \leqslant 1$.
- Two atmospheric plaquettes incident with S are neutral; the remaining are all positive. This requires that $(2d-4)P_+ + 2P_0 \leqslant 1$.
- Finally, one atmospheric plaquette incident with S is negative; the remaining are all positive. This implies that $(2d-3)P_+ + P_- \leqslant 1$.

One of these conditions (the second) is redundant, and the remaining three only provide a restriction on the probabilities. If in addition, one requires that $P_+ = e^{2\beta}P_-$, then in two dimensions the probabilities can be maximized subject to the constraints above to give

$$P_+ = \frac{e^{2\beta}}{1+e^{2\beta}}, \qquad P_- = \frac{1}{1+e^{2\beta}} \qquad \text{and} \qquad P_0 = \frac{1}{2}. \tag{246}$$

This is the optimal implementation of the algorithm in the square lattice.

In three and higher dimensions the situation is more complicated. The relations $(2d-3)P_+ + P_- \leqslant 1$ and $P_+ = e^{2\beta}P_-$ imply that

$$P_+ \leqslant \frac{e^{2\beta}}{1+(2d-3)e^{2\beta}}. \tag{247}$$

At the same time, one must have $(2d-4)P_+ + 2P_0 \leqslant 1$. Maximizing P_+ and solving for P_0 gives $P_0 = (P_+ + P_-)/2$ where $P_+ = e^{2\beta}P_-$. This solution gives the standard choices for the probabilities in $d \geqslant 3$:

$$P_+ = \frac{e^{2\beta}}{1+(2d-3)e^{2\beta}}, \qquad P_- = \frac{1}{1+(2d-3)e^{2\beta}}, \qquad \text{and} \qquad P_0 = \frac{1+e^{2\beta}}{2(1+(2d-3)e^{2\beta})}. \tag{248}$$

With these choices of the transition probabilities, the algorithm can be implemented as follows:

Algorithm 14.1 (The BFACF algorithm).

(1) Initialize the algorithm by choosing the first walk s_0 to be a walk with given (fixed) endpoints x and y. Then the length of the walk is at least the length of the shortest lattice path between x and y. Define the parameter β.
(2) Let s_n be the current walk. With uniform probability $1/|s_n|$, choose an edge S in s_n. Enumerate the $2d-2$ atmospheric plaquettes incident with S.
(3) Execute one of the atmospheric moves such that a particular positive atmospheric move is performed with probability P_+, a particular negative atmospheric move is performed with probability P_- and a particular neutral atmospheric move is performed with probability P_0. The probabilities P_0, P_+ and P_- are given in equation (246) in the square lattice, and in equation (248) in the cubic and hypercubic lattices. This gives a proposed walk s'_n. If the proposed walk is not self-avoiding, then $s'_n = s_n$. If the probabilities P_* sum up to less than one, then $s'_n = s_n$ in the event that no atmospheric move is performed.
(4) Put $s_{n+1} = s'_n$ to find the next state, and continue at step (2) above.

The transition probability of obtaining a walk t from a given walk s, $P_{s\to t}$, is given by

$$P_{s\to t} = \begin{cases} [P_+/|s|], & \text{if} \quad |t| = |s|+2, \\ [P_-/|s|], & \text{if} \quad |t| = |s|-2, \\ [P_0/|s|], & \text{if} \quad |t| = |s|, \end{cases} \tag{249}$$

since the edge S in step (2) is chosen with probability $1/|s|$. These transition probabilities satisfy the condition of detailed balance given by

$$|s|\, P_{s\to t} = |t| P_{t\to s}. \tag{250}$$

This shows that the algorithm samples asymptotically from the distribution

$$P_\beta(s) = \frac{|s| e^{\beta|s|}}{\sum_t |t| e^{\beta|t|}} = \frac{|s| e^{\beta|s|}}{\sum_n n\, c_n(x) e^{\beta n}}, \tag{251}$$

where $c_n(x)$ is the number of self-avoiding walks of length n with endpoints the origin 0 and the fixed lattice site x. This is not the canonical Boltzmann distribution over the state space S_{0x} of walks, and the denominator is finite only if $\beta < -\log\mu$ where μ is the growth constant of walks defined in equation (2).

A simpler implementation of the BFACF-algorithm uses the Metropolis algorithm. In this case steps (2) and (3) above are modified as follows: choose an edge S with uniform probability $1/|s_n|$ in the current walk s_n and translate the edge in a (uniformly chosen) normal direction (there are $2d-2$ possible choices) while inserting or deleting edges to keep the walk connected. If the resulting object is a self-avoiding walk s'_n of length $|s'_n|$ edges, then accept it with probability $\min\{1, e^{\beta(|s'_n|-|s_n|)}\}$ as the next state s_{n+1}. Otherwise, put $s_{n+1} = s_n$. In this implementation one can show that the condition of detailed balance is given by equation (250), and if $\beta < -\log\mu$, then the algorithm samples asymptotically from the distribution $P_\beta(s)$ in equation (251).

Further generalizations of the BFACF algorithm are available in [119] and its generalization to a dynamic Monte Carlo algorithm for sampling ribbons in the cubic lattice appeared in [118].

The mean value of an observable $\mathcal{O}$ over the realization of a Markov chain by the algorithm is given by

$$\langle\langle \mathcal{O} \rangle\rangle_\beta = \frac{\sum_s |s|\, \mathcal{O}(s) e^{\beta|s|}}{\sum_s |s|\, e^{\beta|s|}}. \tag{252}$$

Estimates of canonical Boltzmann mean values can be determined by the ratio estimate

$$\langle \mathcal{O} \rangle_\beta = \frac{\langle\langle \mathcal{O}/|s| \rangle\rangle_\beta}{\langle\langle 1/|s| \rangle\rangle_\beta}. \tag{253}$$

Since the algorithm is ergodic with detailed balance condition in equation (250) over the state space S_{0x} of walks with fixed endpoints 0 and x in two dimensions and $\|x\|_\infty \geqslant 2$ in $d \geqslant 3$, this ratio estimate convergences to $\langle \mathcal{O} \rangle_\beta$ in probability with the length of the realized Markov chain.

The mean length of walks realized by the algorithm depends on β and is given by $\langle\langle |s| \rangle\rangle_\beta$. Since $c_n = \mu^{n+o(n)}$, if follows that as β approaches $-\log\mu$, the longer the average length of walks sampled by this algorithm become. This leads to a critical slowing down as $\beta \nearrow -\log\mu$, and arguments in [16, 100] suggest that the integrated autocorrelation time diverges as $\tau \geqslant C \langle |s| \rangle^{4\nu}$ where ν is the metric exponent of the model and C is a constant. In the square lattice $4\nu = 3$ and in the cubic lattice $4\nu \approx 2.4$.

Generalizations of the BFACF-algorithm can be made by adding other atmospheric moves to the collection of plaquette atmospheric moves. These may include neutral pivot atmospheric moves, which do lead to an improvement in the critical slowing down of the algorithm [118, 119]. Generally, additions of alternative positive or negative atmospheric moves will follow the arguments outlined in [11], or can be done simply by using a Metropolis-style rejection technique.

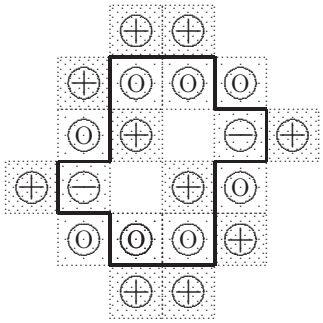

Figure 33. The plaquette atmospheres of a polygon. Positive atmospheric plaquettes are denoted by a +, negative atmospheric plaquettes by − and neutral atmospheric plaquettes by O. In this example, $a_+^p = 10$, $a_0^p = 8$ and $a_-^p = 2$.

15. Monte Carlo simulations of lattice polygons

In this section, we consider briefly the application of the algorithms in the previous sections to lattice polygons.

In bipartite lattices such as the hypercubic lattice in two and larger dimensions, all polygons have even lengths. The existence of a growth constant μ is demonstrated by equations (4) and (5), and these limits exist only if they are taken through even values of n. The scaling of p_n for even values of n is given by equation (25).

As in the case of walks, Monte Carlo simulation of lattice polygons relies on elementary moves which one may define in terms of polygon atmospheres—this was set out in [78]. The implementation of a Monte Carlo algorithm proceeds by first selecting a set of elementary moves, and then by defining dynamics which implements the elementary moves to sample polygons.

If S is a collection of polygons, then a set of elementary moves is *irreducible* on S if for any two members of S communicates with one another along a sequence of elementary moves. Irreducibility is a tricky issue in the cubic lattice for some algorithms, since polygons may be non-trivial knots, and this topological constraint may cut S into distinct ergodicity classes.

We proceed by defining a number of polygon atmospheres which we shall use in constructing algorithms. These atmospheres are by no means exhaustive, and may be mixed or adapted or even generalized to enhance the sampling or to design even more general algorithms.

(1) *Plaquette atmospheres in polygons.* Positive, neutral and negative atmospheric plaquettes can be defined on a polygon in exactly the same way as they were defined for walks. A ⊔-conformation of three edges in a polygon is a *negative atmospheric plaquette*. If an edge in a polygon can be replaced by three edges in a ⊔-conformation to obtain a new polygon, then a *positive atmospheric plaquette* has boundaries the given edge together with the three edges in the ⊔-conformation. Two adjacent edges at 90° with one another and bounding a unit square with exactly two edges and three vertices in the polygon is a *neutral atmospheric plaquette*.

Atmospheric plaquettes on a polygon are indicated in figure 33. The number of positive atmospheric plaquettes is denoted by a_+^p and this is the size of the positive

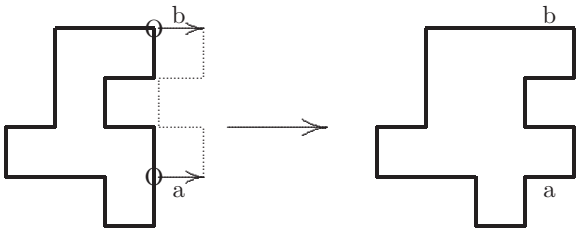

Figure 34. The positive generalized atmosphere of a polygon. Two parallel edges (a and b) are incident on a pair of vertices (marked by O) on the left and the part of the polygon between these vertices is translated one step in the direction of a and b (this is indicated by the dotted line). If the resulting object on the right is a polygon, then the pair (a, b) is part of the *generalized atmosphere* of the polygon, and we call the pair (a, b) a pair of *positive generalized atmospheric edges* of the polygon on the left. The positive generalized atmosphere of the polygon is the set of all such pairs of edges. Negative generalized atmospheric edges are defined as the opposite of the positive generalized atmospheric edges: by contracting the pair (a, b) in the right-hand polygon, the polygon on the left is obtained. Thus, (a, b) is a pair of negative atmospheric edges in this polygon. The negative generalized atmosphere of the polygon is the set of all such pairs of edges. A neutral generalized atmosphere can also be defined in a similar way by contracting one edge and adding a second to produce a new polygon of the same length.

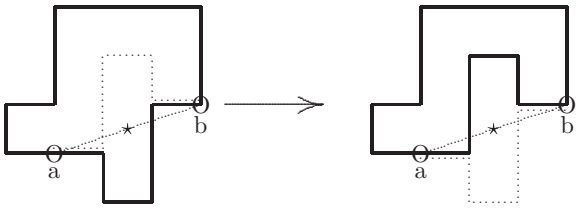

Figure 35. A pivot move in a lattice polygon. The segment of the polygon between the pivots a and b is reflecting through the centre $\star$ of the line segment ab. If the pivots a and b are on the same lattice axis, or on another symmetry line or plane of the lattice, then line and plane reflections, and rotations, from the octahedral group can be used to pivot parts of the polygon. The total collection of pivots on a polygon forms the *neutral pivot atmosphere* of the polygon.

plaquette atmosphere of the polygon. Similarly, a_-^p and a_0^p are the sizes of the negative and neutral plaquette atmospheres.

(2) *Generalized atmospheres in polygons.* Consider two vertices in a polygon, and two parallel edges (a, b) incident with the vertices. An example is given in figure 34. If the polygon is cut into two segments at the vertices, and the pair of edges is inserted, then the outcome may be a lattice polygon. In this case, we call (a, b) a pair of *positive generalized atmospheric edges*. The effect of this construction is that a pair of vertices is replaced by a pair of parallel edges.

Conversely, consider (a, b) to be a pair of parallel edges in a polygon. If these edges are contracted to vertices, and the resulting object is still a polygon, then (a, b) is a pair of *negative generalized atmospheric edges*. A *neutral generalized atmosphere* may be defined by inserting one edge and contracting another.

(3) *Neutral pivot atmospheres for polygons.* A *pivot move* is implemented as follows on a polygon: Choose two pivot points a and b (see figure 35) and note that these cut the polygon into two segments. Choose the shorter of the two segments and operate on it with an element of the lattice symmetry group selected such that the operation

leaves the endpoints of the segment unchanged. If the resulting object is a polygon (thus self-avoiding), then we say that the shorter segment was pivoted about a and b.

Observe that in hypercubic lattices there at least two pivot moves which are always possible; for any choices of the pivoting points a and b: the first is the identity move, and the second is an *inversion*; a point reflection of the segment through the centre of the line segment ab connecting the pivots. In figure 35 an inversion is illustrated.

Other pivots can be performed if a and b are on a line of symmetry or a plane of symmetry of the lattice. If a and b are on the same line L inclined at 45^o with the X-axis in the square lattice, then possible pivot moves include an inversion, and reflections through L and through the bisector of L.

The collection of all possible pivot moves on a polygon forms the *neutral pivot atmosphere* of the polygon.

We next consider four algorithms for polygons. These include two kinetic growth type algorithms (GARM and GAS) [131], and two dynamic algorithms, one canonical (the pivot algorithm, see [28]) and one grand canonical (the BFACF algorithm [4, 8]).

15.1. GARM for polygons in the square lattice

GARM for polygons is implemented similarly to GARM for walks as in section 10, and one can use either plaquette or generalized polygon atmospheres. GARM is then a kinetic growth algorithm for polygons in analogy with the Rosenbluth method (see section 6) and PERM (see section 9) for walks.

In figures 33 and 34, the definitions of plaquette and generalized polygon atmospheres are given, and the implementation of these in a GARM algorithm for polygons will be given below. The plaquette atmospheres induce BFACF-moves on a polygon [4, 8].

Define S to be the state space of all (unrooted) polygons. The smallest polygon is the unit square of length 4 and all polygons in the square lattice will have even length.

The irreducibility of atmospheric moves on S is an important ingredient in the construction of GARM on polygons. For plaquettes atmospheres the following is known:

Theorem 15.1. *The set of BFACF-moves induced by the plaquette atmospheres in figure 33 is irreducible in the square lattice. In particular, by applying negative and neutral plaquette atmospheric moves, any polygon can be reduced to the minimal polygon of length 4 edges in a finite number of steps.*

The proof of this theorem is a special case of the proof of theorem 9.7.2 in [102]. The corollary to the last theorem is that any polygon of length n in the square lattice can be grown from a polygon of length 4 by applying positive and neutral BFACF moves. In other words, a GARM-style algorithm can be defined by growing polygons from the unit square polygon by applying positive and neutral plaquette atmospheric moves. Since every BFACF move is also a generalized atmospheric move in a polygon, it follows that a GARM-style algorithm for square lattice polygons can also be defined by growing polygons with positive and neutral generalized atmospheres [131].

This situation breaks down in the cubic lattice, since knotted conformations obstruct the growing of any knotted polygon by using BFACF or generalized atmospheric moves. Thus, we limit the discussion for the remainder of this section to square lattice polygons.

Consider a polygon s and its associated generalized atmospheric statistics $a_+^g(s)$, $a_0^g(s)$ and $a_-^g(s)$. A new polygon s' can be constructed from s by implementing a positive atmospheric move on s, as in figure 34. In this case s' is a *successor* of s, and $|s'| = |s| + 2$. We call s a *predecessor* of s'.

Consider the following sampling scheme in the state space S of square lattice polygons. Let ϕ_0 be the polygon of length 4 and be the initial state in a sequence ϕ. Generate states (polygons) $\phi_1, \phi_2, \ldots$ by applying positive or neutral atmospheric moves: ϕ_j is obtained from ϕ_{j-1} by a uniformly chosen positive or neutral atmospheric move on ϕ_{j-1}.

This generates a sequence $\phi = \phi_0\phi_1 \cdots \phi_N$ of length $|\phi| = N + 1$ in S where the state ϕ_j is a polygon. Since the state ϕ_j is obtained from ϕ_{j-1} by uniformly selecting a randomly chosen positive or neutral atmospheric move, the conditional probability that ϕ_j follows ϕ_{j-1} is

$$P(\phi_j|\phi_{j-1}) = \frac{1}{a_+^g(\phi_{j-1}) + a_0^g(\phi_{j-1})}. \tag{254}$$

The probability to sample a given sequence ϕ is therefore

$$\Pr(\phi) = \prod_{j=1}^{|\phi|-1} P(\phi_j|\phi_{j-1}) = \prod_{j=1}^{|\phi|-1} \left[\frac{1}{a_+^g(\phi_{j-1}) + a_0^g(\phi_{j-1})}\right]. \tag{255}$$

The final state of the sequence ϕ is a specific state $\phi_N \equiv \tau$, and the probability that a sequence terminates in the state τ is given by

$$\Pr(\tau) = \sum_{\phi\to\tau} \prod_{j=1}^{|\phi|-1} \left[\frac{1}{a_+^g(\phi_{j-1}) + a_0^g(\phi_{j-1})}\right], \tag{256}$$

where the sum is over all sequences ϕ in s which starts as the trivial state ϕ_0 (the square polygon of length four), and with final state τ.

As in the case of GARM for walks, the key to the algorithm is to assign a weight $W(\phi)$ to each sequence: it is appropriate to define this by

$$W(\phi) = \prod_{j=1}^{|\phi|-1} \left[\frac{a_+^g(\phi_{j-1}) + a_0^g(\phi_{j-1})}{a_-^g(\phi_j) + a_0^g(\phi_j)}\right]. \tag{257}$$

Each factor in this product is the ratio of the atmospheres of the current state, divided by the atmospheres of the next state. This choice of the weight leads to lemma 10.1. In this case, it follows that the mean weight of sequences ending in τ is one:

$$\langle W(\phi)\rangle = \sum_{\phi\to\tau} W(\phi) Pr(\phi) = 1 \tag{258}$$

and the proof proceeds as in lemma 10.1. The counting formula in equation (201) generalizes to

$$\sum_{|\tau|=n} \langle W(\phi)\rangle = \sum_{\substack{\phi\to\tau \\ |\tau|=n}} \prod_{j=1}^{|\phi|-1} \left[\frac{1}{a_-^g(\phi_j) + a_0^g(\phi_j)}\right] = p_n. \tag{259}$$

Thus, GARM is again an approximate enumeration algorithm, and can be used to determine p_n.

The implementation of GARM for polygons is similar to the implementation for walks in algorithm 10.2, with the added observation that a trapped sequence cannot occur with either BFACF or generalized atmospheric moves.

Algorithm 15.2 (GARM for polygons).

(1) Let N be the length of the sequence to be generated and put the weight $W = 1$. Let ϕ_0 be the unit square polygon of length 4. Determine the positive and neutral atmospheres of ϕ_j. Since $a_0^g(\phi_0) = 0$, determine ϕ_1 by uniformly selecting a positive atmospheric move and executing it, and put $j = 1$.

(2) Determine (positive, neutral and negative) atmospheres of ϕ_j. Update the weight W by multiplying it with $\left(a_+^g(\phi_{j-1}) + a_0^g(\phi_{j-1})\right)\Big/\left(a_-^g(\phi_j) + a_0^g(\phi_j)\right)$.
(3) Select a positive or neutral atmospheric move uniformly from those available on ϕ_j and construct ϕ_{j+1} by executing it.
(4) Increment $j \to j+1$. If $j < N$, then determine the next state by going to step (2). If $j = N$ then a sequence of length N states and weight W have been generated. Continue by starting a new sequence at step (1) or terminate the algorithm.

This is the simplest implementation of GARM for polygons in the square lattice and it gives a scheme for the kinetic growth and approximate enumeration of polygons.

It is possible that estimates of the weight W along realized sequences ϕ in a simulation will have large variance. This undermines the statistics of the algorithm and similar to PERM, a pruning and enrichment step in the algorithm may be needed to improve the sampling. Eventually, this approach will give a flat histogram version of GARM for square lattice polygons (flatGARM), as described in section 10.

A thermal implementation of GARM is similarly obtained by computing estimates of the partition function at an activity $z = e^{\beta}$:

$$Z_n^{\text{est}}(z) = \frac{1}{N}\sum_{k=1}^{N} W\left(\phi_j^k\right) z^{E(\phi_j^k)} = \langle W z^E \rangle_j, \tag{260}$$

where ϕ_j^k is the jth state of length n in the kth sequence ϕ^k. In algorithm 10.3 the calculation of r is replaced by

$$r = \frac{W(\phi_j) z^{E(\phi_j)}}{\langle W z^E \rangle_j} \tag{261}$$

for state ϕ_j. The rest of the implementation is unchanged. This will enrich and prune states such that a flat histogram over the partition function at activity z is obtained. An alternative approach to this is to implement a microcanonical version of GARM as in algorithm 10.4.

15.1.1. GARM for polygons in three dimensions. The implementation of GARM for polygons in three dimensions (the cubic lattice) is at this time not resolved and remains an open question. The selection of atmospheric moves similar to plaquette and generalized atmospheres for cubic lattice polygons seems unproblematic, but proving that any polygon can be reached via a sequence of positive and neutral atmospheric moves (as was done in theorem 15.1) remains unresolved, even if the model is restricted polygons in the class of unknotted polygons. This problem can be solved by adding neutral pivot atmospheres into the problem—since these moves are known to be irreducible on polygons of fixed length [99], it is then possible to prove that positive and neutral atmospheres can be used to generate any polygon in the cubic lattice, starting from the unit square. Unfortunately, computing pivot neutral atmospheres is computationally expensive, and leads to a significant reduction in the effectiveness of such a proposed algorithm.

15.2. Generalized atmospheric sampling of polygons

Generalized atmospheric sampling on the atmospheres of polygons can similarly be implemented to sample polygons along a sequence ϕ. This method generalizes GARM by implementing negative atmospheric moves in addition to neutral and positive atmospheric moves.

The implementation follows a general outline similar to walks. Consider a polygon s together with its associated atmospheric statistics $a_+^g(s)$, $a_0^g(s)$ and $a_-^g(s)$ of positive, neutral and negative atmospheres. A new polygon s' can be constructed from s by selecting a positive, neutral or negative atmospheric move, and then constructing s' by implementing the atmospheric move by addition or deletion of edges, or by changing the conformation of s via a neutral atmospheric move. In this case, a *successor* s' is obtained; we call s the *predecessor* of s'. The rules for selecting an atmospheric move will be explained below.

Let ϕ_0 be an initial state in the algorithm. We say that ϕ_0 is in level 0. Successors of ϕ_0 are states ϕ_1 in level 1 which can be reached from ϕ_0 by implementing a (positive, neutral or negative) atmospheric move. The state ϕ_0 in level 0 is also the predecessor of the states ϕ_1 in level 1, and the relationship between predecessors and successors is graphically illustrated in figure 28.

Similarly, successors of a state ϕ_j in level j are determined by considering all states in level $j+1$ which can be obtained from ϕ_j by implementation of an atmospheric move. In this way, states ϕ_j can be sampled in level j by recursively selecting a successor of the current state starting at the state ϕ_0. ϕ_0 is the *source state* at level 0 of the sequence $\phi = \phi_0\phi_1, \ldots, \phi_N$ of length N.

The general implementation now follows the steps in GAS for walks, by selecting a set of atmospheric moves. The BFACF moves are suitable in the square lattice since they are irreducible on the set of all polygons in the square lattice [102]. In three dimensions, the BFACF moves are suitable for the GAS-sampling of polygons of fixed knot type in the cubic lattice. This follows from the next theorem:

Theorem 15.3. *The irreducibility classes of the BFACF moves for unrooted polygons in the cubic lattice coincide with the knot types of the polygons as closed simple curves in three dimensions.*

The proof of theorem 15.3 can be found in [80]. The algorithm proceeds by sampling states (polygons) in a sequence in the state space S. The state ϕ_{j+1} is obtained from ϕ_j by implementing an atmospheric move as follows: introduce the parameter β (possibly dependent on the length of state ϕ_j) and perform atmospheric moves with probabilities

$$P_+(\text{positive atmospheric move}) = \frac{\beta a_+^g(\phi_j)}{a_-^g(\phi_j) + a_0^g(\phi_j) + \beta a_+^g(\phi_j)}, \tag{262}$$

$$P_0(\text{neutral atmospheric move}) = \frac{a_0^g(\phi_j)}{a_-^g(\phi_j) + a_0^g(\phi_j) + \beta a_+^g(\phi_j)}, \tag{263}$$

$$P_-(\text{negative atmospheric move}) = \frac{a_-^g(\phi_j)}{a_-^g(\phi_j) + a_0^g(\phi_j) + \beta a_+^g(\phi_j)}, \tag{264}$$

where in particular β_j is dependent on the level j, and may be updated between realizations of sequences, and where these probabilities are normalized to sum up to unity.

A maximum length on the polygons may be imposed on the algorithm by *defining* the positive atmosphere of polygons of length $n_{\max}$ to be zero. In this event, irreducibility of the algorithm (if its atmospheric moves include BFACF-moves) follows from theorem 15.1.

Generally, the average length of polygons in the sequence may be controlled by the parameter β. If one fixes

$$\beta_\ell = \frac{\langle a_-^g(\phi)\rangle_\ell}{\langle a_+^g(\phi)\rangle_\ell} \tag{265}$$

Table 4. Approximate 2D polygon enumeration with GAS.

n	c_n	Estimate
4	1	1
6	2	1.999 367
8	7	6.995 924
10	28	27.996 66
12	124	123.958
14	588	587.738
16	2938	2937.38
18	15 268	15 276.0
20	81 826	81 881.0
22	449 572	449 554
24	2521 270	$2.519\,19 \times 10^{6}$
26	14 385 376	$1.436\,33 \times 10^{7}$
28	83 290 424	$8.307\,37 \times 10^{7}$
30	488 384 528	$4.864\,34 \times 10^{8}$
32	2895 432 660	$2.882\,31 \times 10^{9}$
34	17 332 874 364	$1.725\,18 \times 10^{10}$
36	104 653 427 012	$1.041\,98 \times 10^{11}$
38	636 737 003 384	$6.333\,37 \times 10^{11}$
40	3900 770 002 646	$3.871\,527 \times 10^{12}$

for each state ϕ of length ℓ, then the probability of stepping to the next level via a positive atmospheric step (and increasing the length of the polygon) is on average equal to the probability of stepping to the next level via a negative atmospheric step (and decreasing the length of the walk). This change will produce a flat histogram on the length ℓ of the states sampled by GAS in the interval $(4, 6, \ldots, n_{\max})$. This defines the flatGAS algorithm on polygons and its implementation is given by algorithm 11.2.

In table 4, estimates of p_n obtained by an implementation of flatGAS using plaquette atmospheres are given. In this simulation, $n_{\max} = 40$ while 50 sequences of length 5000 000 each were generated, and average weights were computed to determine estimates of p_n.

15.3. The pivot and cut-and-paste algorithms for polygons

The pivot algorithm for polygons uses a neutral pivot atmospheric move (illustrated in figure 35) as an elementary move. The algorithm is implemented by uniformly choosing two vertices as pivots, and then attempting a pivot by choosing a pivot move from those possible with uniform probability. If the resulting object is self-avoiding, then it is accepted by the algorithm, otherwise it is rejected and the current polygon is again read as the next state.

The rotations and reflections of parts of the polygon between two pivots are randomly chosen from the symmetry group of the lattice, but are constrained by the orientation of the pivots. Each such operation must leave the pivots unchanged or reflect them into one another. In this context, there is at least one elementary move which can be performed for arbitrary oriented pivots: this is the *inversion*, or the reflection of a segment of the polygon through the origin which is located at the centre of mass of the pivots: a vertex with coordinates $(X, Y, \ldots, Z)$ is reflected to $(-X, -Y, \ldots, -Z)$. This is the move illustrated in figure 35.

Other possible elementary moves include reflections through and rotations about lattice axes, reflections through lattice planes normal to the displacement vector between pivots, or through places which contains the pivots, and reflections and rotations about axes and planes inclined at 45° with lattice axes, and so forth. In d dimensions, there are $2^d d!$ possible moves, but at every step only those which keep the pivots invariant will be considered as possible moves.

The irreducibility properties of the pivot algorithm in d dimensions are well understood. We give this in the following theorem due to Madras *et al* [99].

Theorem 15.4. *The pivot algorithm for polygons is irreducible if the set of elementary moves includes inversions, rotations by* 90° *about lattice axes, and reflections through the line bisecting the vector between the pivot points.*

The implementation of the algorithm is as follows:

Algorithm 15.5 (The pivot algorithm for polygons).

(1) Let s_1 be the initial polygon of length n.
(2) Suppose that s_j is the current polygon. Choose two pivots v_1 and v_2 uniformly in s_j.
(3) Determine the shorter subwalk P between the two pivots and determine the vector $\mathbf{V} = v_2 - v_1$.
(4) Perform a pivot (selected with uniform probability from a list of possible pivots) on P such that v_1 and v_2 are invariant or are reflected into one another, subject to the following:
   (a) If $\mathbf{V}$ is parallel to a line inclined at 45° with a lattice axis, then include a reflection through the line bisecting $\mathbf{V}$ and an inversion in the list of possible pivots.
   (b) If $\mathbf{V}$ is parallel to a lattice axis, then include a rotation through 90° about $\mathbf{V}$ and an inversion in the list of possible moves.
   (c) Otherwise, include an inversion (a point reflection through the midpoint of $\mathbf{V}$ in the list of possible moves.
(5) If the resulting object is a polygon, then take it as the state s_{j+1}. Otherwise it has a self-intersection and $s_{j+1} = s_j$.
(6) Increment j by 1 and stop if the time series of sampled polygons is long enough. Otherwise, continue by starting at step (2).

The pivot algorithm is very efficient in sampling polygons in the canonical (fixed length) ensemble. It can be implemented via the Metropolis algorithm to sample polygons in an interacting model. Umbrella sampling and multiple Markov chain Monte Carlo sampling follow the same implementation as those algorithms are implemented with the pivot algorithm for walks.

In figure 36, a time series generated by the pivot algorithm on square lattice polygons is displayed. The size of the negative plaquette atmosphere of the polygons where tracked every 30 attempted pivots. The series is correlated, but time series analysis can be used effectively to determine means and variances of such data.

A generalization of the pivot algorithm for polygons is obtained as follows: instead of selecting two pivots, select M pivots and cut the polygon into M subwalks. Reflect or rotate each subwalk such that its endpoints remain fixed or are reflected into one another. Shuffle the subwalks and attempt to reconnect the M subwalks into a polygon. Since the displacement vector between the endpoints of each subwalk fixed, putting the subwalks end-to-end still closes the walks into a ring. If this is self-avoiding, then the polygon is accepted as the next state by the algorithm, otherwise it is rejected and the original polygon is read again as the next state.

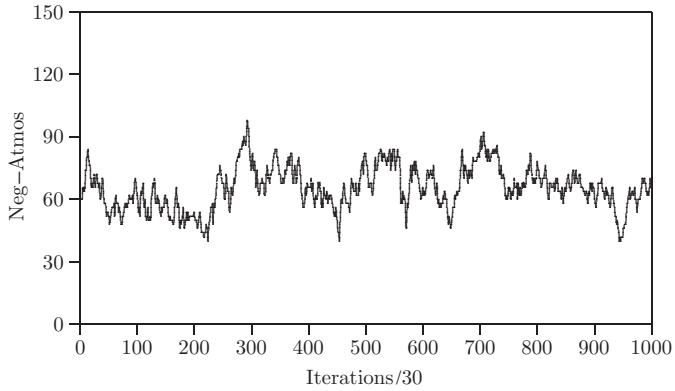

Figure 36. Time series data from a simulation of polygons of length $n = 300$ using the pivot algorithm. Plotted is the size of the negative plaquette atmospheric statistic. The data are correlated and time series analysis can be used to determine means and variances.

Fast implementations of the pivot algorithm for polygons follow from the work of Clisby in [18]. In this case, the basic pivot move on polygons of length n can be performed in $O(\log n)$ CPU-time units. This implementation of the pivot algorithm should enable accurate simulations of extremely long self-avoiding polygons.

15.4. The BFACF algorithm

The BFACF algorithm for polygons is implemented by applying the BFACF atmospheric moves to a polygon. The implementation of the algorithm is similar to the BFACF algorithm for walks with fixed endpoints in algorithm 14.1. The only difference is that the polygon is not rooted at endpoints, as the walks were, but is free to float in the lattice as the BFACF algorithm performs elementary moves on it.

It is known that this algorithm is irreducible in the square lattice [102], but the irreducibility properties are more complicated in the cubic lattice. The following theorem [80] resolves the issue in the cubic lattice.

Theorem 15.6. *The ergodicity classes of the BFACF-algorithm for unrooted polygons in the cubic lattice coincide with the knot types of the polygons as closed simple curves in three dimensions.*

Since the ergodicity classes coincide with the knot types of the polygons, this result implies that knotted polygons of a fixed knot type can be simulated by the BFACF algorithm (see for example [71, 74, 78, 81]). The first polygon in the Markov chain determines the knot type of the polygons in the simulation, and the sampling is uniform at each fixed length n while the distribution over n is given by the stretched Boltzmann distribution first derived for the BFACF algorithm and walks in section 14. In the case of knotted polygons, this distribution is given by

$$P_\beta(s, K) = \frac{|s|e^{\beta|s|}}{\sum_n n\ p_n(K)e^{\beta n}} \tag{266}$$

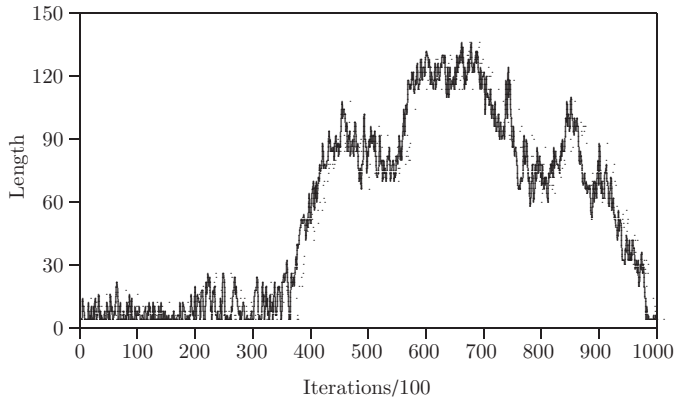

Figure 37. Time series data on two-dimensional polygons using the BFACF algorithm. Plotted is the length of the polygon at every 100th iteration. In this simulation $e^{\beta} = 0.377 < 1/\mu = 0.379\ldots$. Increasing β typically leads to increased correlations, but even in this display the data have large correlations and long series must be generated to determine means and variances.

in the cubic lattice, where $p_n(K)$ is the number of (unrooted) polygons of length n and knot type K. In the square lattice, and in hypercubic lattices with dimensions bigger than three, the algorithm is irreducible and the distribution is

$$P_\beta(s) = \frac{|s|e^{\beta|s|}}{\sum_n n\, p_n e^{\beta n}}, \tag{267}$$

where p_n is the number of (unrooted) polygons of length n. Estimates of canonical Boltzmann mean values can be determined by the ratio estimate in equation (253).

Critical slowing down occurs in this algorithm as $\beta \nearrow 1/\mu$, and arguments in [102] suggest that the integrated autocorrelation time diverges as $\tau \geqslant C \cdot \langle |s| \rangle^{4\nu}$ where ν is the metric exponent of the model and C is a constant. In the square lattice $4\nu = 3$ and in the cubic lattice $4\nu \approx 2.4$.

In figure 37, the length of a two-dimensional square lattice polygon is plotted in a time series in a BFACF simulation with $e^{\beta} = 0.377$. This time series is quite typical of these simulations; polygons tend to stay close to their minimal length 4 for extended periods, while periodically making an excursion to longer lengths before returning. The distribution of the excursions bounds the autocorrelation time of the time series, and if e^{β} is close to $1/\mu$, then very long series is needed to determine the means and variances of measured quantities. Reducing β also reduces autocorrelations, but long polygons are then only rarely sampled. Various schemes are used to improve the sampling at longer lengths while keeping autocorrelations under control—in particular, mixing the BFACF algorithm with pivot moves improves the situation; see for example results in [120, 121, 146].

16. Conclusions

There has been significant progress in the Monte Carlo simulation of lattice walks and polygons (and related structures such as trees and animals) since the invention of the Rosenbluth algorithm [132] in 1955. Big advances in this field include the invention of the scanning method [105], the BFACF [4, 8] and pivot algorithms [92, 102] in the 1980s, and PERM [44]

in the 1990s. The extension of the Rosenbluth method and PERM [44] to GARM [131] and GAS [79] since 2000 is in all probability not the last words in this area, and even more efficient algorithms are still to be found.

In contrast, the exact enumeration of walks (see for example [83] and Iwan Jensen's website) remains a viable and effective alternative strategy for numerical work on walks and polygons. While the best numerical estimates for exponents still comes from exact series analysis, that process is inherently exponential in computational cost. Since Monte Carlo errors tend to decrease as a power law with increasing computational effort, one would expect Monte Carlo simulations to eventually become competitive with series results. This may not be practically so for two reasons. The first is that series has an astonishing advantage in the current accuracy (μ is known to ten digits in two dimensions [84, 85], while Monte Carlo simulations give five digits [112]). The second reason is that by clever implementation of series techniques, the exponential reckoning of the method may be pushed back arbitrarily (but finitely) far.

Monte Carlo simulations for walks and polygons are on the other hand very useful in determining phase diagrams of interacting models of walks and polygons as models of interacting polymers, in particular when implemented in a multiple Markov chain or microcanonical style.

Overall, the general area of numerical simulations of walks and polygons remains very active and is further stimulated by the availability since the mid-1980s of cheaper and faster computing power. Moore's law and more efficient algorithms, in collaboration with the many areas in physics and mathematics which intersect this field, will keep the field active for the foreseeable future.

Acknowledgments

EJJvR is grateful for support in the form of a Discovery Grant from NSERC (Canada). In addition, E Orlandini provided data for figures 9 and 10 and J Alvarez provided data for figures 5, 6 and 20.

References

[1] Alexandrowicz Z 1969 Monte Carlo of chains with excluded volume: a way to evade sample attrition *J. Chem. Phys.* **51** 561–5
[2] Alexandrowicz Z 1983 Self-avoiding walks of continuous spatial dimensionality *Phys. Rev Lett* **50** 736–9
[3] Alvarez J, Janse van Rensburg E J and Rechnitzer A 2009 Phase transitions in genelarized atmospheres for self-avoiding walks *Preprint*
[4] Aragão de Carvalho C, Caracciolo S and Fröhlich J 1983 Polymers and $g|\phi|^4$-theory in four dimensions *Nucl. Phys.* B **215** 209–48
[5] Bachmann M and Janke W 2003 Multicanonical chain-growth algorithm *Phys. Rev. Lett.* **91** 208105–8
[6] Barber M N 1973 Scaling relations for critical exponents of surface properties of magnets *Phys. Rev.* B **8** 407–9
[7] Barber M N, Guttmann A J, Middlemiss K M, Torrie G M and Whittington S G 1978 Some tests of scaling theory for a self-avoiding walk attached to a surface *J. Phys. A: Math. Gen.* **11** 1833–42
[8] Berg B and Foester D 1981 Random paths and random surfaces on a digital computer *Phys. Lett.* **106B** 323–6
[9] Berretti A and Sokal A D 1985 New Monte Carlo method for the self-avoiding walk *J. Stat. Phys.* **40** 483–531
[10] Brak R, Guttmann A J and Whittington S G 1991 On the behaviour of collapsing linear and branched polymers *J. Math. Chem.* **8** 255–67
[11] Brak R and Nidras P P 1997 New Monte Carlo algorithm for interacting self-avoiding walks *J. Phys. A: Math. Gen.* **30** 1457–69
[12] Burkhardt T W, Eisenriegler E and Guim I 1989 Conformal theory of energy correlations in the semi-infinite two dimensional $O(n)$ model *Nucl. Phys.* B **316** 559–72

[13] Cardy J L and Saleur H 1989 Universal distance ratios for two dimensional polymers *J. Phys. A: Math. Gen.* **22** L601–L604
[14] Caracciolo S, Causo M S and Pelissetto A 1998 High-precision determination of the critical exponent gamma for self-avoiding walks *Phys. Rev.* E **57** 1215–18
[15] Caracciolo S, Guttmann A J, Jensen I, Pelissetto A, Rogers A N and Sokal A D 2004 Correction-to-scaling exponent for two-dimensional self-avoiding walks *J. Stat. Phys.* **120** 1037–100
[16] Caracciolo S, Pelissetto A and Sokal A D 1990 Non-local Monte Carlo algorithm for self-avoiding walks with fixed endpoints *J. Stat. Phys.* **60** 1–53
[17] Chang I S and Meirovitch H 1993 Collapse transition of self-avoiding walks on a square lattice in bulk and near a linear wall: the universality classes of the Θ and Θ' points *Phys. Rev.* E **48** 3656–60
[18] Clisby N 2008 An accurate estimate of v for self-avoiding walks via the pivot algorithm *Preprint* (Submitted to *Phys. Rev. Lett.*)
[19] Clisby N, Liang R and Slade G 2007 Self-avoiding walk enumeration via the Lace expansion *J. Phys. A: Math. Theor.* **40** 10973–1017
[20] Coniglio N, Jan N, Majid I and Stanley H E 1988 Conformation of a polymer at the θ' point: connection to the external perimeter of a percolation cluster *Phys. Rev.* B **35** 3617–20
[21] Daoud M and de Gennes P G 1977 Statistics of macromolecular solutions trapped in small pores *J. Physique* **38** 85–93
[22] Dayantis J and Palierne J-F 1991 Monte Carlo precise determination of the end-to-end distribution function of self-avoiding walks on the simple-cubic lattice *J. Chem. Phys.* **95** 6088–99
[23] Dayantis J and Palierne J-F 1996 Scaling exponents of the self-avoiding walk problem in three dimensions *Phys. Rev.* B **49** 3217–25
[24] Dayantis J and Palierne J-F 1996 Correction to scaling exponents for self-avoiding walks in two dimensions *Macromol. Theory Simul.* **5** 1143–50
[25] de Gennes P-G 1975 Collapse of a polymer chain in poor solvents *J. Physique* **36** L55–L57
[26] de Gennes P-G 1976 Scaling theory of polymer adsorption *J. Physique* **37** 1445–52
[27] de Gennes P-G 1979 *Scaling Concepts in Polymer Physics* (Ithaca, NY: Cornell University Press)
[28] Dubins L E, Orlitsky A, Reeds J A and Shepp L A 1988 Self-avoiding random loops *IEEE Trans. Inf. Theory* **34** 1509–16
[29] Duplantier B 1986 Polymer network of fixed topology: renormalization, exact critical exponent γ in two dimensions *Phys. Rev. Lett.* **57** 941–4
[30] Duplantier B 1986 Tricritical polymer chains in or below three dimensions *Europhys. Lett.* **1** 491–8
[31] Duplantier B 1987 Geometry of polymer chains near the θ-point and dimensional regularisation *J. Chem. Phys.* **86** 4233–44
[32] Duplantier B 1990 Renormalisation and conformal invariance for polymers *Fundamental Problems in Statistical Mechanics VII* ed H van Beijeren (Amsterdam: Elsevier) pp 171–223
[33] Duplantier B 1993 Reply: new scaling form for the collapse polymer phase *Phys. Rev. Lett.* **71** 4274–74
[34] Duplantier B and Saleur H 1987 Exact tricritical exponents for polymers at the Θ-point in two dimensions *Phys. Rev. Lett.* **59** 539–42
[35] Duplantier B and Saleur H 1988 Reply to: universality classes for the θ and θ' points *Phys. Rev. Lett.* **60** 1204
[36] Edwards S F 1965 The statistical mechanics of polymers with excluded volume *Proc. Phys. Soc. Lond.* **85** 613–24
[37] Eizenberg N and Klafter J 1993 Self-avoiding walks on a simple cubic lattice *J. Chem. Phys.* **99** 3976–82
[38] Flory P J 1949 The configuration of a real polymer chain *J. Chem. Phys.* **17** 303–10
[39] Flory P J 1969 *Statistical Mechanics of Chain Molecules* (New York: Wiley-Interscience)
[40] Foster D, Orlandini E and Tesi M C 1992 Surface critical exponents for models of polymer collapse and adsorption: the universality of the θ- and θ'-points *J. Phys. A: Math. Gen.* **25** 1211–17
[41] Geyer C J and Thompson E A 1995 Annealing Markov chain Monte Carlo with applications to ancestral inference *J. Am. Stat. Assoc.* **90** 909–20
[42] Gilks W R and Spiegelhalter D J 1996 *Markov Chain Monte Carlo in Practice* (Boca Raton, FL: CRC Press)
[43] Grassberger P 1993 Monte Carlo simulations of 3D self-avoiding walks *J. Phys. A: Math. Gen.* **26** 2769–76
[44] Grassberger P 1997 Pruned-enriched Rosenbluth method: simulation of θ-polymers of chain length up to 1000 000 *Phys. Rev.* E **56** 3682–93
[45] Grassberger P and Hegger R 1995 Comment on: surface critical exponents of self-avoiding walks on a square lattice with adsorbing linear boundary: a computer simulation study *Phys. Rev.* E **51** 2674–76
[46] Grassberger P and Hegger R 1995 Simulations of three-dimensional θ-polymers *J. Chem. Phys.* **102** 6881–99
[47] Grassberger P and Hsu H-P 2002 Stretched polymers in a poor solvent *Phys. Rev.* E **65** 031807-1–031807-9
[48] Guida R and Zinn-Justin J 1998 Critical exponents of the N-vector model *J. Phys. A: Math. Gen.* **31** 8103–21

[49] Guim I and Burkhardt T W 1989 Transfer-matrix study of the adsorption of a flexible self-avoiding polymer chain in two dimensions *J. Phys. A: Math. Gen.* **22** 1131–40
[50] Guim I and Burkhardt T W 1994 Crossover exponent for polymer adsorption in two dimensions *Phys. Rev.* E **49** 1495–99
[51] Guttmann A J 1975 On the Recurrence relation method of series analysis *J. Phys. A: Math. Gen.* **5** 1081–88
[52] Guttmann A J 1987 On the critical behaviour of self-avoiding walks *J. Phys. A: Math. Gen.* **20** 1839–54
[53] Guttmann A J 1989 On the critical behaviour of self-avoiding walks: II *J. Phys. A: Math. Gen.* **22** 2807–13
[54] Guttmann A J and Conway A R 2001 Square lattice self-avoiding walks and polygons *Ann. Comb.* **5** 319–45
[55] Guttmann A J and Whittington S G 1978 Self-avoiding walks in a slab of finite thickness: a model of steric stabilisation *J. Phys. A: Math. Gen.* **5** L107–L10
[56] Hammersley J M 1960 Limiting properties of numbers of self-avoiding walks *Phys. Rev.* **118** 656
[57] Hammersley J M 1961 The number of polygons on a lattice *Math. Proc. Camb. Phil. Soc.* **57** 516–23
[58] Hammersley J M 1962 Generalisation of the fundamental theorem on sub-additive functions *Math. Proc. Camb. Phil. Soc.* **58** 235–8
[59] Hammersley J M and Morton K W 1954 Poor man's Monte Carlo *J. R. Stat. Soc.* B **16** 23–38
[60] Hammersley J M, Torrie G M and Whittington S G 1982 Self-avoiding walks interacting with a surface *J. Phys. A: Math. Gen.* **15** 539–71
[61] Hammersley J M and Welsh D J A 1962 Further results on the rate of convergence to the connective constant of the hypercubic lattice *Q. J. Math. Oxford* **13** 108–10
[62] Hara T and Slade G 1990 The Lace expansion for self-avoiding walks in five or more dimensions *Rev. Math. Phys.* **4** 235–327
[63] Hara T and Slade G 1991 Critical behaviour of self-avoiding walks in five or more dimensions *Bull. Am. Math. Soc.* **25** 417–23
[64] Hara T, Slade G and Sokal A 1993 New lower bounds on the self-avoiding-walk connective constant *J. Stat. Phys.* **72** 479–517
[65] Hastings W K 1970 Monte Carlo sampling methods using Markov Chains and their applications *Biometrika* **57** 97–109
[66] Hegger R and Grassberger P 1994 Chain polymers near an adsorbing surface *J. Phys. A: Math. Gen.* **27** 4069–81
[67] Hsu H-P and Grassberger P Collapsed two-dimensional polymers on a cylinder *J. Phys. A: Math. Gen.* **35** L759–L766
[68] Hsu H-P, Nadler W and Grassberger P 2004 Scaling of star polymers with 1–80 arms *Macromolecules* **37** 4658–63
[69] Ioffe D and Velenik Y 2008 Ballistic phase of self-interacting self-avoiding walks *Analysis and Stochastics of Growth Processes and Interface Models* ed P Mörters *et al* (Oxford: Oxford University Press) pp 55–79
[70] Janse van Rensburg E J 1992 Ergodicity of the BFACF algorithm in three dimensions *J. Phys. A: Math. Gen.* **25** 1031–42
[71] Janse van Rensburg E J 2007 Squeezing knots *J. Stat. Mech.* P03001
[72] Janse van Rensburg E J, Orlandini E, Owczarek A L, Rechnitzer A and Whittington S G 2005 Self-avoiding walks in a slab with attractive walls *J. Phys. A: Math. Gen.* **38** L823–L828
[73] Janse van Rensburg E J, Orlandini E and Whittington S G 2006 Self-avoiding walks in a slab: rigorous results *J. Phys. A: Math. Gen.* **39** 13869–902
[74] Janse van Rensburg E J, Orlandini E, Sumners D W, Tesi M C and Whittington S G 1996 The writhe of knots in the cubic lattice *J. Knot. Theory Ram.* **6** 31–44
[75] Janse van Rensburg E J, Orlandini E, Tesi M C and Whittington S G 2008 Knotting in stretched polygons *J. Phys. A: Math. Theor.* **41** 015003
[76] Janse van Rensburg E J, Orlandini E, Tesi M C and Whittington S G 2009 Thermodynamics and entanglements of walks under stress *J. Stat. Mech.* P07014
[77] Janse van Rensburg E J and Rechnitzer A 2004 Multiple Markov Chain Monte Carlo study of adsorbing self-avoiding walks in two and in three dimensions *J. Phys. A: Math. Gen.* **37** 6875–98
[78] Janse van Rensburg E J and Rechnitzer A 2008 Atmospheres of polygons and knotted polygons *J. Phys. A: Math. Theor.* **41** 105002
[79] Janse van Rensburg E J and Rechnitzer R 2009 Generalized atmospheric sampling of self-avoiding walks *J. Phys. A: Math. Theor.* to be published
[80] Janse van Rensburg E J and Whittington S G 1991 The BFACF algorithm and knotted polygons *J. Phys. A: Math. Gen.* **24** 5553–67
[81] Janse van Rensburg E J and Whittington S G 1991 The dimensions of knotted polygons *J. Phys. A: Math. Gen.* **24** 3935–48
[82] Jensen I 2003 A parallel algorithm for the enumeration of self-avoiding polygons on the square lattice *J. Phys. A: Math. Gen.* **36** 5731–45

[83] Jensen I 2004 Enumeration of self-avoiding walks on the square lattice *J. Phys. A: Math. Gen.* **37** 5503–24
[84] Jensen I and Guttmann A J 1998 Self-avoiding walks, neighbour-avoiding walks and trials on semi-regular lattices *J. Phys. A: Math. Gen.* **31** 8137–45
[85] Jensen I and Guttmann A J 1999 Self-avoiding polygons on the square lattice *J. Phys. A: Math. Gen.* **32** 4867–76
[86] Kennedy T 2002 A faster implementation of the Pivot algorithm for self-avoiding walks *J. Stat. Phys.* **106** 407–29
[87] Kennedy T 2002 Monte Carlo tests of SLE predictions for the 2D self-avoiding walk *Phys. Rev. Lett.* **88** 130601–5
[88] Kennedy T 2005 Monte Carlo comparisons of the self-avoiding walk and SLE as parameterized curves arXiv:math.PR/0510604
[89] Kesten H 1963 On the number of self-avoiding walks *J. Math. Phys.* **4** 960–9
[90] Kesten H 1964 On the number of self-avoiding walks: II *J. Math. Phys.* **5** 1128–37
[91] Krawczyk J, Prellberg T, Owczarek A L and Rechnitzer A 2004 Stretching of a chain polymer adsorbed at a surface *J. Stat. Mech.* P10004
[92] Lal M 1969 Monte Carlo computer simulation of chain molecules I *Mol. Phys.* **17** 57–64
[93] Lawler G, Schramm O and Werner W 2004 On the scaling limit of planar self-avoiding walk In *Fractal Geometry and Applications: A Jubilee of Benoit Mandelbrot* vol II ed M Lapidus and M van Frankenhuijsen (Providence, RI: American Mathematical Society) pp 339–64
[94] Lawrie I D and Sarlbach S 1984 Tricriticality *Phase Transitions and Critical Phenomena* vol 9 ed C Domb and J L Lebowitz (New York: Academic) pp 65–161
[95] Le Guillou J C and Zinn-Justin J 1980 Critical exponents from field theory *Phys. Rev.* B **21** 3976–98
[96] Le Guillou J C and Zinn-Justin J 1985 Accurate critical exponents from the ϵ-expansion *J. Physique Lett.* **46** L137–L141
[97] Le Guillou J C and Zinn-Justin J 1985 Accurate critical exponents from field theory *J. Physique* **50** 1365–70
[98] Li B, Madras N and Sokal A D 1995 Critical exponents, hyperscaling, and universal amplitude ratios for two and three dimensional self-avoiding walks *J. Stat. Phys.* **80** 661–754
[99] Madras N, Orlitsky A and Shepp L A 1990 Monte Carlo generation of self-avoiding walks with fixed endpoints and fixed length *J. Stat. Phys.* **58** 159–83
[100] Madras N and Slade G 1993 *The Self-Avoiding Walk* (Basle: Birkhäuser)
[101] Madras N and Sokal A D 1987 Nonergodicity of local, length-preserving Monte Carlo algorithms for the self-avoiding walk *J. Stat. Phys.* **47** 573–95
[102] Madras N and Sokal A D 1988 The Pivot algorithm: a highly efficient Monte Carlo method for the self-avoiding walk *J. Stat. Phys.* **50** 109–86
[103] Maes D and Vanderzande C 1990 Self-avoiding rings at the θ-point *Phys. Rev.* A **41** 3074–80
[104] Masand B, Wilensky U, Massar J P and Redner S 1992 An extension of the two-dimensional self-avoiding walks series on the square lattice *J. Phys. A: Math. Gen.* **25** L365–9
[105] Meirovitch H 1985 Scanning method as an unbiased simulation technique and its application to the study of self-attracting random walks *Phys. Rev.* A **32** 3699–708
[106] Meirovitch H and Chang I 1993 Surface critical exponents of self-avoiding walks on a square lattice with an adsorbing boundary *Phys. Rev.* E **48** 1960–69
[107] Meirovitch H and Lim H A 1989 The collapse transition of self-avoiding walks on a square lattice: a computer simulation study *J. Chem. Phys.* **91** 2544–54
[108] Meirovitch H and Livne S 1988 Computer simulation of long polymers adsorbed on a surface: II. critical behaviour of a single self-avoiding walk *J. Chem. Phys.* **88** 4507–15
[109] Metropolis N and Ulam S 1949 The Monte Carlo method *J. Am. Stat. Assoc.* **44** 335–41
[110] Metropolis N, Rosenbluth A W, Rosenbluth M N, Teller A H and Teller E 1953 Simulated annealing *J. Chem. Phys.* **23** 1087–92
[111] Napper D H 1983 *Polymeric Stabilization of Colloidal Dispersions* (London: Academic)
[112] Nidras P 1996 Grand canonical simulations of the interacting self-avoiding walk model *J. Phys. A: Math. Gen.* **29** 7929–42
[113] Nidras P and Brak R 1997 New Monte Carlo algorithms for interacting self-avoiding walks *J. Phys. A: Math. Gen.* **30** 1457–69
[114] Nienhuis B 1982 Exact critical point and critical exponents on $O(n)$ models in two dimensions *Phys. Rev. Lett.* **49** 1062–65
[115] Nienhuis B 1984 Coulomb gas formulation of the two-dimensional phase transitions *Phase Transitions and Critical Phenomena* vol 11 ed C Domb and J L Lebowitz (New York: Academic) pp 1–53
[116] Nienhuis B 1984 Critical behavior of two-dimensional spin models and charge asymmetry in the coulomb gas *J. Stat. Phys.* **34** 731–61

[117] O'Brien G L 1990 Monotonicity of the number of self-avoiding walks *J. Stat. Phys.* **59** 969–79
[118] Orlandini E, Janse van Rensburg E J and Whittington S G 1996 A Monte Carlo algorithm for lattice ribbons *J. Stat. Phys.* **82** 1159–98
[119] Orlandini E, Tesi M C, Whittington S G, Sumners D W and Janse van Rensburg E J 1994 The writhe of a self-avoiding walk *J. Phys. A: Math. Gen.* **27** L333–8
[120] Orlandini E, Tesi M C, Janse van Rensburg E J and Whittington S G 1997 The shapes of self-avoiding polygons with torsion *J. Phys. A: Math. Gen.* **30** L693–8
[121] Orlandini E, Tesi M C, Janse van Rensburg E J and Whittington S G 1998 Asymptotics of knotted lattice polygons *J. Phys. A: Math. Gen.* **31** 5953–67
[122] Owczarek A L, Prellberg T and Brak R 1993 New scaling form for the collapsed polymer phase *Phys. Rev. Lett.* **70** 951–53
[123] Owczarek A L, Prellberg T and Brak R 1993 Reply: new scaling form for the collapsed polymer phase *Phys. Rev. Lett.* **71** 4275
[124] Pincus P 1976 Excluded volume effects and stretched polymer chains *Macromolecules* **9** 386–8
[125] Poole P H, Coniglio N, Jan N and Stanley H E 1988 Universality classes for the θ and θ' points *Phys. Rev. Lett.* **60** 1203
[126] Prellberg T and Krawczyk J 2004 Flat histogram version of the pruned and enriched Rosenbluth method *Phys. Rev. Lett.* **92** 120602–5
[127] Prellberg T and Owczarek A L 2000 Four-dimensional polymer collapse: pseudo-first-order transition in interacting self-avoiding walks *Phys. Rev.* E **62** 3780–9
[128] Privman V 1986 Study of the θ-point by enumeration of self-avoiding walks on the triangular lattice *J. Phys. A: Math. Gen.* **19** 3287–97
[129] Rapaport D C 1985 On three-dimensional self-avoiding walks *J. Phys. A: Math. Gen.* **18** 113–26
[130] Rechnitzer A and Janse van Rensburg E J 2002 Canonical Monte Carlo determination of the connective constant of self-avoiding walks *J. Phys. A: Math. Gen.* **35** L605–12
[131] Rechnitzer A and Janse van Rensburg E J 2008 Generalized atmospheric Rosenbluth methods (GARM) *J. Phys. A: Math. Theor.* **41** 442002
[132] Rosenbluth M N and Rosenbluth A W 1955 Monte Carlo calculation of the average extension of molecular chains *J. Chem. Phys.* **23** 356–9
[133] Saleur H 1986 Collapse of two-dimensional linear polymers *J. Stat. Phys.* **45** 419–38
[134] Saleur H 1987 Conformal invariance for polymers and percolation *J. Phys. A: Math. Gen.* **20** 455–60
[135] Schramm O 2001 Scaling limits of loop-erased random walks and uniform spanning trees *Israel J. Math.* **118** 221–88
[136] Seno F and Stella A L 1988 θ-point of a linear polymer in 2 dimensions: a renormalization group analysis of Monte Carlo enumerations *J. Physique* **49** 739–48
[137] Seno F and Stella A L 1988 Surface exponents for a linear polymer at the $d = 2\,\Theta$ point *Europhys. Lett.* **7** 605–10
[138] Seno F, Stella A L and Vanderzande C 1988 Universality class of the $d = 2\,\theta$-point of linear polymers *Phys. Rev. Lett.* **61** 1520
[139] Sokal A D 1996 Monte Carlo methods for the self-avoiding walk *Nucl Phys.* B **47** 172–9
[140] Sokal A D and Thomas L E 1989 Exponential convergence to equilibrium for a class of random walk models *J. Stat. Phys.* **54** 797–828
[141] Sumners D W and Whittington S G 1988 Knots in self-avoiding walks *J. Phys. A: Math. Gen.* **21** 1689–94
[142] Sun S-T, Nishio I, Swislow G and Tanaka T 1980 The coilglobule transition: radius of gyration of polystyrene in cyclohexane *J. Chem. Phys.* **73** 5971–75
[143] Symanzik K 1969 Euclidean quantum field theory *Local Quantum Theory* ed R Jost (New York: Academic)
[144] Tesi M C, Janse van Rensburg E J, Orlandini E and Whittington S G 1995 Interacting self-avoiding walks and polygons in three dimensions *J. Phys. A: Math. Gen.* **29** 2451–63
[145] Tesi M C, Janse van Rensburg E J, Orlandini E and Whittington S G 1996 Monte Carlo study of the interacting self-avoiding walk model in three dimensions *J. Stat. Phys.* **82** 155–81
[146] Tesi M C, Janse van Rensburg E J, Orlandini E and Whittington S G 1997 Torsion of polygons in $\mathbb{Z}^3$ *J. Phys. A: Math. Gen.* **30** 5179–94
[147] Torrie G M and Valleau J P 1977 Nonphysical sampling distributions in Monte Carlo free-energy estimation: umbrella sampling *J. Comput. Phys.* **23** 187–99
[148] Torrie G M and Whittington S G 1975 Exact enumeration of neighbour-avoiding walks on the tetrahedral and body-centred cubic lattices *J. Phys. A: Math. Gen.* **5** 1178–84
[149] Verdier P H and Stockmayer W H 1962 Monte Carlo calculations on the dynamics of polymers in dilute solution *J. Chem. Phys.* **36** 227–35

[150] Vanderzande C, Stella A L and Seno F 1991 Percolation, the special θ' point, and the θ-θ' universality puzzle *Phys. Rev. Lett.* **67** 2757–60
[151] Veal A R, Yeomans J M and Jug G 1991 The effect of attractive monomer–monomer interactions on adsorption of a polymer chain *J. Phys. A: Math. Gen.* **24** 827–50
[152] Vrbova T and Prochazka K 1999 Adsorption of self-avoiding walks at an impenetrable plane in the expanded phase: a Monte Carlo study *J. Phys. A: Math. Gen.* **32** 5469–75
[153] Wall F T and Erpenbeck J J 1959 Statistical computation of radii of Gyration and mean internal dimensions of polymer molecules *J. Chem. Phys.* **30** 634–7