Generalized atmospheric sampling of self-avoiding walks

IOP PUBLISHING JOURNAL OF PHYSICS A: MATHEMATICAL AND THEORETICAL

J. Phys. A: Math. Theor. **42** (2009) 335001 (29pp) doi:10.1088/1751-8113/42/33/335001

Generalized atmospheric sampling of self-avoiding walks

E J Janse van Rensburg[1] **and A Rechnitzer**[2]

[1] Department of Mathematics and Statistics, York University Toronto, Ontario M3J 1P3, Canada
[2] Department of Mathematics, The University of British Columbia, Vancouver, British Columbia V6T 1Z2, Canada

E-mail: rensburg@yorku.ca and andrewr@math.ubc.ca

Received 12 March 2009, in final form 17 June 2009
Published 28 July 2009
Online at stacks.iop.org/JPhysA/42/335001

Abstract

In this paper, we introduce a new Monte Carlo method for sampling lattice self-avoiding walks. The method, which we call 'GAS' (generalized atmospheric sampling), samples walks along weighted sequences by implementing elementary moves generated by the positive, negative and neutral atmospheric statistics of the walks. A realized sequence is weighted such that the average weight of states of length n is proportional to the number of self-avoiding walks from the origin c_n. In addition, the method also self-tunes to sample from uniform distributions over walks of lengths in an interval $[0, n_{\max}]$. We show how to implement GAS using both generalized and endpoint atmospheres of walks and analyse our data to obtain estimates of the growth constant and entropic exponent of self-avoiding walks in the square and cubic lattices.

PACS numbers: 82.35.Lr, 05.50.+q, 05.40.Fb, 64.60.De

1. Introduction

Polymer statistics and enumeration are a classical problem in polymer physics which has been modelled by lattice self-avoiding walks and related objects [6, 8, 9, 10]. The self-avoiding walk is a standard model of polymer enumeration, and it has been analysed using scaling theory, numerical approaches and field theory, see for example [1, 7, 12, 13, 21].

Monte Carlo sampling of self-avoiding walk models of polymers has long been a key activity in the study of walks as models of polymers [1–3, 25, 26]. Numerous ingenious algorithms have been used over the last 50 years to sample walks; these include the Rosenbluth algorithm [25], the BFACF algorithm [1, 2], the Berretti–Sokal algorithm [3], the pivot algorithm [20], the pruned and enriched Rosenbluth method (PERM) [11] and the incarnation of PERM as generalized atmospheric pruned and enriched Rosenbluth sampling (GARM) [24].

In this paper, we propose a general algorithm for the approximate enumeration of self-avoiding walks by Monte Carlo sampling. We call this algorithm 'GAS', for 'generalized atmospheric sampling' (of walks). The algorithm is a generalization of GARM, and it operates by sampling walks kinetically in the same spirit as the Rosenbluth, PERM and GARM algorithms, but now with steps that reduce the length of the walk added into the mix of possible moves. Similar to Rosenbluth sampling (and also to PERM and GARM), GAS will be an approximate enumeration scheme—sequences of weighted walks are sampled such that their average weights at a given length n are proportional to the number of walks of length n.

We define c_n to be the number of self-avoiding walks of length n from the origin in the hypercubic lattice. For example, $c_0 = 1$ in all dimensions d, while in the square lattice $c_1 = 4$, $c_2 = 12$, $c_3 = 36$ and so on. The *growth constant* of self-avoiding walks is defined by the limit [13]

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

and this also shows that c_n growth exponentially with n: it is generally thought that

$$c_n \sim n^{\gamma-1}\mu^n \tag{2}$$

where γ is the *entropic exponent* of self-avoiding walks.

The algorithm we propose in this paper (GAS) will sample walks along sequences with average weights proportional to c_n. More precisely, GAS will sample a sequence of states (walks) $\phi = \langle \phi_0, \phi_1, \phi_2, \ldots, \phi_j, \ldots, \phi_L\rangle$ of weight $W(\phi)$, starting from an arbitrary source state ϕ_0 (this will usually be the trivial walk of length 0). The average weight of a sequence of states starting from state ϕ_0 and terminating in the state τ will be denoted by $\langle W(\phi)\rangle_\tau$ and will be defined later.

The implementation of the algorithm, starting from the trivial state ϕ_0, will be such that

$$\frac{\sum_{|\tau|=n}\langle W(\phi)\rangle_\tau}{\sum_{|\sigma|=m}\langle W(\phi)\rangle_\sigma} \to \frac{c_n}{c_m} \tag{3}$$

as the length of the sequences goes to infinity, and where the summations are over all possible final states τ and σ in the sequences ϕ such that τ is a walk of length n and σ is a walk of length m.

GAS will estimate the left-hand side averages in equation (3), and so will give an estimate of the ratio on the right-hand side. If one chooses $m = 0$, then $c_m = 1$ and the ratio of the weights on the left-hand side is a direct estimate of c_n. In this way, GAS will be an approximate enumeration algorithm like the Rosenbluth method, and the sequences of walks it samples can also be analysed to compute averages of other observables such as metric quantities.

In section 2, we describe the basic ideas underlying GAS. We explain the sampling of the algorithm in terms of paths in a graph we call the *derivative graph*; this notion has its origin in the GARM algorithm [24], but the sampling in the case of GAS is more general.

In section 3, we discuss the elementary moves of GAS in terms of self-avoiding walk atmospheres [23, 24]. We explain that these moves are irreducible in the sense that every two walks in the state space of walks are connected to each other by a finite sequence of elementary moves. That is, the state space of walks together with the set of atmospheric moves constitutes a connected graph.

In section 4, we describe the particular implementations of GAS in this paper. We focus on the two sets of atmospheric moves examined in section 3, but we also note that other definitions for atmospheres can be used. Each choice of a set of atmospheric moves gives rise to a different version of GAS, and in this paper we focus only on the two sets of atmospheres defined in section 3; the first implementation is with generalized atmospheric moves [24], while the second implementation uses endpoint atmospheres [23].

We explain the implementation of a flat-histogram version of GAS in section 5. The flat-histogram sampling is achieved by the appropriate (self)-tuning of the single parameter of the GAS algorithm. This implementation has the advantage that sampling is asymptotically a flat-histogram distribution over the lengths of the walk, and we achieve this result without the implementation of enrichment and pruning moves as in flatPERM [22] and GARM [24]. The flat-histogram sampling is a natural consequence of the implementation of the GAS algorithm. This kind of sampling enables one to collect statistics over sets of walks of given lengths while controlling for the statistical errors in the sample over the entire region of interest.

The implementation of a flat-histogram version of GAS with generalized atmospheric moves is a generalization of GARM by the addition of negative and neutral atmospheric moves in the set of elementary moves of the GARM algorithm. The implementation of GAS with endpoint atmospheric moves is a generalization of PERM [11, 22] and of the Beretti–Sokal algorithm [3], and we call this implementation ‘GABS’ to distinguish it from the GAS implementation with generalized atmospheric moves.

In section 5, we present numerical results of simulations using GAS and GABS. We examine the properties of GAS for walks of lengths $n \in [0, 249]$ in two and three dimensions. The average weights of sequences give approximate estimates of c_n and we use extrapolations of ratio estimators to estimate μ and γ in equation (2) in two and three dimensions. We obtain the estimates $\mu = 2.6383 \pm 0.0002$ and $\gamma = 1.34 \pm 0.02$ in two dimensions, and $\mu = 4.684 \pm 0.001$ and $\gamma = 1.16 \pm 0.02$ in three dimensions. The values of μ are consistent with results found elsewhere (μ is estimated in two dimensions by series enumeration in [15] and by using the lace expansion to generate series in three dimensions in [5]). The estimates for γ is close to the expected exact value of the entropic exponent in two dimensions ($\gamma = 43/32 = 1.343\,75$ [7, 21]), and in three dimensions ($\gamma = 1.160 \pm 0.004$ [16–18]).

Simulations using the flat-histogram GABS version of GAS proved more efficient because the implementation of endpoint atmospheric moves is computationally fast. This enabled us to sample walks of lengths in the interval [0, 999] in two and three dimensions. Analysing the results gives the estimates $\mu = 2.6383 \pm 0.0001$ and $\gamma = 1.34 \pm 0.02$ in two dimensions and $\mu = 4.684 \pm 0.001$ while $\gamma = 1.16 \pm 0.02$. Comparison of these results to those obtained using GAS gives best estimates for μ and γ:

$$\mu = 2.6383 \pm 0.0001, \qquad \text{if} \quad d = 2, \qquad \text{and} \quad \mu = 4.684 \pm 0.001, \qquad \text{if} \quad d = 3, \tag{4}$$

for the growth constant in two and three dimensions. These results are consistent with estimates for μ from exact enumeration studies in two and three dimensions, namely $\mu = 2.638\,15\ldots$ in $d = 2$ [15] and $\mu = 4.6840\ldots$ in $d = 3$ [5].

For the entropic exponent,

$$\gamma = 1.34 \pm 0.02, \qquad \text{if} \quad d = 2, \qquad \text{and} \quad \gamma = 1.16 \pm 0.02, \qquad \text{if} \quad d = 3. \tag{5}$$

These results are consistent with the exact value of $\gamma = 43/32$ in two dimensions [7, 21], and with the estimate $\gamma = 1.160 \pm 0.004$ [16–18] obtained by other methods.

We conclude the paper with a few remarks in section 7. In particular, we examine alternative statistics for estimating μ by tracking ratios of atmospheric statistics. Our results are comparable to the results given above.

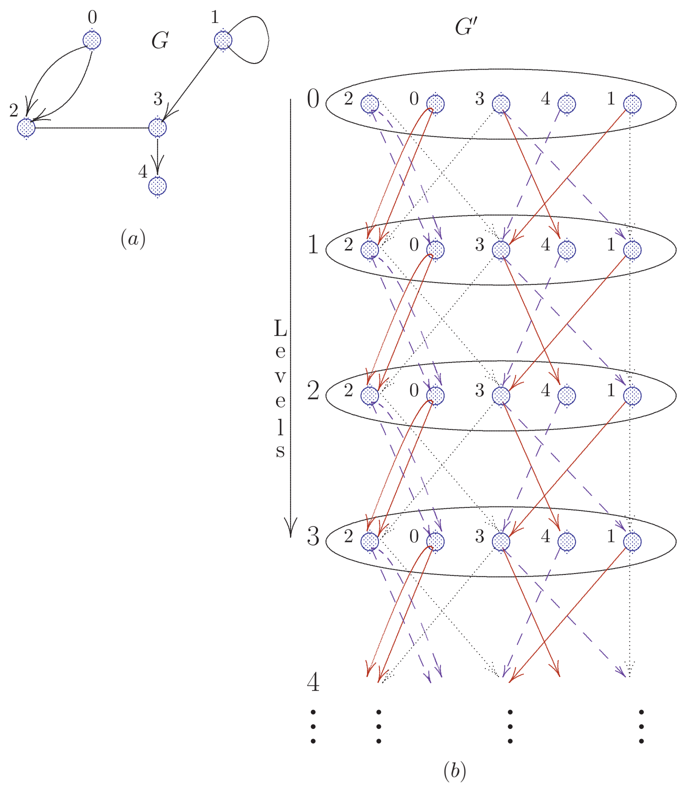

Figure 1. (*a*) The graph on the left has five states (vertices) labelled {0, 1, 2, 3, 4}. (*b*) Vertices in the graph G on the left has now been arranged in rows and copied into levels labelled 0, 1, 2, ... as illustrated, where each level is delineated by an ellipse and labelled from the top down. Directed arcs $\overrightarrow{ab}$ in G are added as arcs $\underline{a_\ell b_{\ell+1}}$ (with a solid arrow) and as $\overrightarrow{b_\ell a_{\ell+1}}$ (with a dashed arrow) in G'. Undirected edges $\overline{ab}$ are added as arcs $\overrightarrow{a_\ell b_{\ell+1}}$ and $\overrightarrow{b_\ell a_{\ell+1}}$ (with dotted lines). A (undirected) loop $\overline{aa}$ in G is added as an arc $a_\ell a_{\ell+1}$.

2. Atmospheres and derivative graphs

Consider an arbitrary multi-graph G with both directed and undirected edges (for example, the graph in figure 1(*a*)). G has n vertices and may be infinite (in which case $n = \infty$). The vertices in G will also be called the 'states' of the graph.

A vertex $v \in G$ has an incident set of edges $A(v) \subseteq E(G)$. For example, the vertex labelled 3 in G in figure 1 has $A(3) = \{\overrightarrow{13}, \overrightarrow{34}, \overline{23}\}$ (one outgoing directed edge, one incoming directed edge and one undirected edge). We will refer to this set $A(v)$ as the *atmosphere* of v. Observe that the atmosphere of a vertex is a multi-set, for example, the atmosphere of the vertex labelled by 0 in figure 1(*a*) has $A(0) = \{\overrightarrow{02}, \overrightarrow{02}\}$; that is, the arc $\overrightarrow{02}$ occurs twice and the cardinality of $A(0)$ is $|A(0)| = 2$.

The atmosphere of a vertex is further subdivided into disjoint sets $A(v) = P(v) \cup N(v) \cup Z(v)$ where the sets $P(v)$, $N(v)$ and $Z(v)$ are defined by

$$P(v) = \{\vec{e} \in E(G)|\, \vec{e} \text{ is a directed edge starting at } v\}, \tag{6}$$

$$N(v) = \{\vec{e} \in E(G)|\, \vec{e} \text{ is a directed edge ending at } v\}, \tag{7}$$

$$Z(v) = \{\bar{e} \in E(G)|\, \bar{e} \text{ is an undirected edge incident to } v\}. \tag{8}$$

We call $P(v)$ the *positive atmosphere* of v, $N(v)$ the *negative atmosphere* of v and $Z(v)$ the *neutral atmosphere* of v.[3] Note that $\overrightarrow{vw} \in P(v)$ if and only if $\overrightarrow{vw} \in N(w)$. Similarly $\overline{vw} \in Z(v)$ if and only if $\overline{vw} \in Z(w)$. Hence we require that loops in G are always undirected and are part of the neutral atmosphere of a vertex (see figure 1).

The *size* of the atmospheres of a vertex is the cardinality of the multi-sets $P(v)$, $N(v)$ and $Z(v)$. For example, in figure 1(a) the size of the positive atmosphere of vertex 2 is $|P(2)| = 0$, of the negative atmosphere is $|N(2)| = 2$ and of the neutral atmosphere is $|Z(2)| = 1$.

Generalized atmospheric sampling is implemented by performing a random walk on a graph such as G. Since G is a generalized multi-graph with edges and arcs, we make the definition of a random walk in it precise by considering a *derivative graph* G' in figure 1(b)—random walks in G will correspond to directed paths in G' and so be given a precise definition.

We construct the derivative graph as follows: the states or vertices in G are arranged in horizontal rows as indicated in figure 1(b). Each row is a copy of the vertices in G, and we label the rows by a *level number* ℓ. The top row of vertices is in level $\ell = 0$, the second row is in level $\ell = 1$, and so on. Vertices in G' are labelled first the label of their corresponding vertex in G and then by level number. For example, the vertex label 2 in G will correspond to vertex 2_ℓ in level ℓ in G'.

Directed arcs are now inserted between levels in G' so that each arc points from a vertex in level ℓ to a vertex in level $\ell + 1$. That is, the arcs impose a general direction on the vertices from the top down in the derivative graph G' as illustrated in figure 1(b). More precisely

- If $\overrightarrow{ab}$ is an arc in G, then for each $\ell = 0, 1, 2, \ldots,$ we insert the arc $\overrightarrow{a_\ell b_{\ell+1}}$ in G'. These arcs are denoted by solid-line arrows in figure 1(b).
- If $\overrightarrow{ba}$ is an arc in G, then for each $\ell = 0, 1, 2, \ldots,$ we insert the arc $\overrightarrow{a_\ell b_{\ell+1}}$ in G' (note the reversal of direction). These arcs are denoted by long dashed-line arrows in figure 1(b).
- If $\overline{ab}$ is an (undirected) edge in G, then for each $\ell = 0, 1, 2, \ldots,$ we insert the arcs $\overrightarrow{a_\ell b_{\ell+1}}$ and $\overrightarrow{b_\ell a_{\ell+1}}$ in G'. These arcs are denoted by dotted-line arrows in figure 1(b). If $a \equiv b$ then the edge is an (undirected) loop in G, and we insert one arc $a_\ell a_{\ell+1}$ in G'. These arcs are indicated by dotted-line arrows $a_\ell a_{\ell+1}$ in figure 1(b).

The resulting derivative graph is a digraph with source vertices in level $\ell = 0$. We note that the derivative graph is a directed graph with the property that each vertex in level $\ell > 0$ has indegree equal to its outdegree.

We define a *random walk* in the multi-graph G as an initial vertex followed by a sequence of consecutive arcs or edges, where edges and arcs can be traversed in either direction. If a step in the random walk is along a multi-edge or multi-arc, then one of the available multi-edges or multi-arcs is chosen uniformly as the step.

Such a random walk in G is represented by a directed path in the derivative graph G' which descends one level with each step. For example, the walk $0202311\ldots$ in G corresponds to the directed path $0_0 2_1 0_2 2_3 3_3 1_4 1_5 \ldots$ in G'. Each step in the random walk in G' is selected randomly from some distribution from the available (outgoing) arcs to the next level in the atmosphere of the current vertex.

The GAS algorithm will be best understood as the sampling of states in G by realizing a weighted directed path in the derivative graph G' by starting at a (source) state in level $\ell = 0$

[3] These definitions of the terms positive, negative and neutral atmospheres should be seen as a generalization of the use of these terms in [24]. In particular, the edges in the positive atmosphere of a vertex represent constructions that increase the size of the object represented by that vertex—such as appending a bond to the end of a self-avoiding walk. Similarly, edges in the negative atmosphere of a vertex represent constructions that decrease the size of the object—such as deleting the last bond of a self-avoiding walk. Finally, edges in the neutral atmosphere represent size-preserving constructions, such as a rotation of the final bond in a self-avoiding walk.

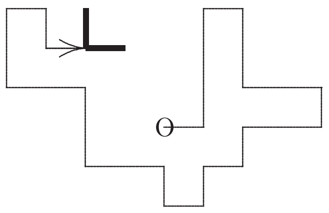

Figure 2. The positive endpoint atmosphere of this walk is composed of the bold edges incident with the last vertex of the walk. The size of the positive endpoint atmosphere in this example is $a_e^+ = 2$.

and then stepping down levels from state to state along arcs. This dynamics is related to the dynamics of the GARM algorithm, but now with the generalization that negative atmospheric steps can be taken by the algorithm (while in GARM, only positive and neutral atmospheric steps are available). This generalization will extend the basic sampling approach in GARM to a wider collection of models, including polygons of fixed knot type (which cannot be sampled using GARM).

In the following section, we explain GAS in terms of the derivative graph and show that it is an approximate enumeration algorithm for the number of states in a graph.

3. Atmospheric moves in self-avoiding walks

In this section, we define a set of elementary moves called *atmospheric moves* on self-avoiding walks. There are numerous possible definitions, but we shall consider only two in this paper, consult references [14, 23, 24] for more about the atmospheres of self-avoiding walks.

3.1. Endpoint atmospheres

In figure 2, the *endpoint atmosphere* of a self-avoiding walk is illustrated. The set of edges incident with the final vertex of the walk composes the *positive endpoint atmosphere* of the walk. The *size* of the positive endpoint atmosphere is the number of edges which can be appended to the last vertex of the walk to create a new self-avoiding walk. We denote the size of the positive endpoint atmosphere by a_+^e. In figure 2, the bold edges compose the positive endpoint atmosphere of the walk with last vertex indicated the arrowhead. In this example, $a_+^e = 2$.

The *negative endpoint atmosphere* of a walk is its terminal edge. By deleting this edge, a self-avoiding walk is obtained, of length reduced by one. The size of the negative endpoint atmosphere is denoted by a_-^e, and it follows that $a_-^e(s) = 0$ if s is the walk of length zero, and $a_-^e(s) = 1$ otherwise. In figure 3, the bold final edge in the walk composes the negative atmosphere. Removing it gives a self-avoiding walk of length reduced by one step.

A *neutral endpoint atmosphere* can also be defined for a self-avoiding walk. Consider the walk in figure 3. The two alternative positions for the final bold step are indicated by dashed line segments. These form the neutral endpoint atmosphere of the walk. The size of the neutral endpoint atmosphere of a walk is denoted by a_0^e, and in the example in figure 3, $a_0^e = 2$.

Endpoint atmospheres can be used to sample walks along a sequence. In figure 4, we show that by starting with the trivial walk as a seed, by the addition of edges from the positive atmosphere at each step, an arbitrary walk can be created. These additions of edges from the

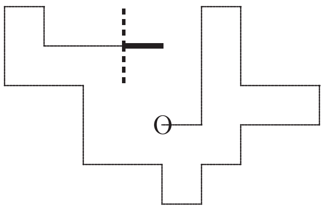

Figure 3. The negative endpoint atmosphere is composed of the bold final edge in this walk. The neutral endpoint atmosphere of this walk is composed of the dashed edges which are alternative positions for the terminal edge in the walk. In this example $a_0^e = 2$.

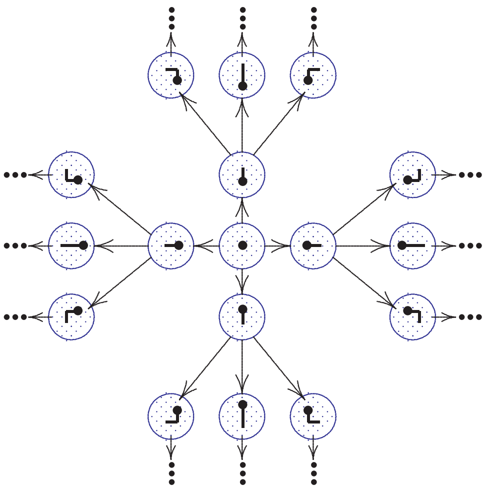

Figure 4. Starting at the trivial walk in the middle, a sequence of positive endpoint atmospheric moves is a walk in this graph, stepping from state to state along the arrows by adding an edge from the positive endpoint atmosphere at every step.

positive endpoint atmosphere are *positive endpoint atmospheric moves*, and they are denoted by arrows in figure 4. Starting from the origin in the middle of the graph, a path along arrows in the graph is a possible sequence of positive endpoint atmospheric moves.

A negative endpoint atmospheric move is executed if the terminal edge of a walk is removed. Such moves are the opposite of the positive endpoint atmospheric moves illustrated in figure 4. Observe that there is a path from the origin (which is the walk of zero length and one vertex) to any other walk s: by executing negative atmospheric moves on s recursively, one finally ends in the origin in figure 4. The reverse sequence of positive endpoint atmospheric moves is a path from the origin to any walk. The corollary to this is that by executing positive and negative endpoint atmospheric moves, there is a path in the graph in figure 4 between any two given walks. We call the collection of positive and negative atmospheric moves *irreducible* on the set of all self-avoiding walks starting from the origin. This property is not disturbed if neutral endpoint atmospheric moves are added to the set of atmospheric moves.

3.2. Generalized atmospheres

A second set of self-avoiding walk atmospheres we will use in this paper is the *generalized atmospheres*.

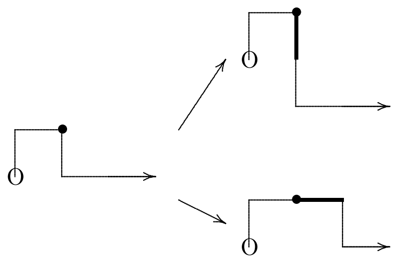

Figure 5. Defining the positive generalized atmosphere of a walk. By inserting the two bold edges at the vertex • in the walk on the left, the walks on the right are obtained. Thus, the bold edges are part of the positive generalized atmosphere of the walk on the left. All edges which can be inserted in this way compose the positive generalized atmosphere of size a_+^g. The walk on the left has $a_+^g = 14$.

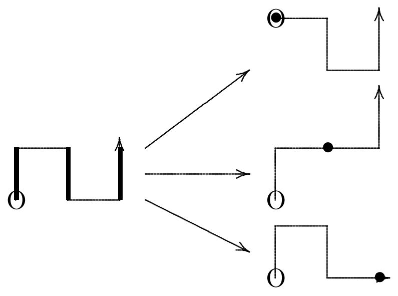

Figure 6. Defining the negative generalized atmosphere of a walk. By deleting and contracting one of the three bold edges in the walk on the left, the walks on the right are obtained. Each bold edge is part of the negative generalized atmosphere of the walk on the left. All the edges which can be deleted and contracted in this way compose the negative generalized atmosphere of size a_-^g. The walk on the left has $a_-^g = 3$.

The definition of a *positive generalized atmosphere* of a given self-avoiding walk is illustrated in figure 5. Cut the walk in a given vertex • to obtain two subwalks. Attempt to reconnect the subwalks by adding a single edge between them, as illustrated. If the resulting walk is self-avoiding then the added edge is part of the positive generalized atmosphere of the walk. For example, the walk on the left of figure 5 has two edges in its positive generalized atmosphere incident with the vertex •, these are denoted by the bold edges on the right.

The size of the positive generalized atmosphere is denoted by a_+^g. If s is the walk on the left in figure 5, then $a_+^g(s) = 14$.

The definition of a *negative generalized atmosphere* for a given self-avoiding walk is illustrated in figure 6. By contracting anyone of the bold edges, a self-avoiding walk of length reduced by one is obtained. The collection of edges which can be contracted to obtain a walk of length reduced by one composes the negative generalized atmosphere of a given self-avoiding walk. The walk on the left in figure 6 has three edges in its negative generalized atmosphere which therefore has size $a_-^g = 3$. Observe that for all walks, except the trivial walk of length zero, the final edge is also a negative generalized atmospheric edge, in other words, $a_-^g(s) \geqslant 1$ if s is a walk of length at least one.

A *neutral generalized atmosphere* can also be defined for walks. One such definition would be based on *pivot moves* [19] which are implemented by choosing a vertex (pivot point) which cuts the walk into two parts, followed by rotating or reflecting the shorter segment around the pivot point in one of the $2^d d!$ possible elements of the symmetry group of the

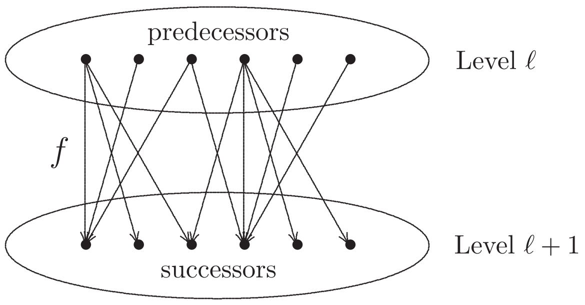

Figure 7. Generalized atmospheric moves in the GAS algorithm set up a map $f : S \to S$ which maps predecessors in S to successors in S. f may be represented as a digraph as illustrated above. Observe that the outdegree of any predecessor vertex in S is equal to the indegree of the corresponding successor vertex. This is necessarily true, since every atmospheric move is reversible. That is, if $\omega \in S$ is a state, then $\mathrm{indeg}\,\omega = \mathrm{outdeg}\,\omega$, since every arc into ω is also an arc out of ω.

d-dimensional hypercubic lattice. The move is a neutral atmospheric move if the result is a self-avoiding walk. The neutral generalized atmosphere is the total collection of distinct pivot moves on a given walk s, denoted by $a_0^g(s)$. For example, if s is the self-avoiding walk of length 2 edges both stepping in the east direction, then $a_0^g(s) = 2d - 2$ in the d-dimensional hypercubic lattice.

Positive, neutral and negative generalized atmospheres similarly define atmospheric moves on a walk, and the result is again similar to figure 4. Walks may be represented as vertices or states in a graph with arcs representing positive generalized atmospheric moves. Since endpoint atmospheric moves are a subset of generalized atmospheric moves, the collection of generalized atmospheric moves is irreducible on the state space of all self-avoiding walks: any walk can be transformed into any other walk by a sequence of generalized atmospheric moves.

Observe that atmospheric moves are reversible. Each positive atmospheric move which increases the length of a walk can be reversed by a corresponding negative atmospheric move. A neutral atmospheric move is similarly reversible by a neutral atmospheric move.

There are alternative definitions for self-avoiding walk atmospheres, see for example references [14, 24], and these may be implemented in this paper as well. However, we shall only consider the two cases defined above in our numerical simulations.

4. Generalized atmospheric sampling of walks

In this section, we use the ideas presented in the previous two sections to explain the implementation of a Monte Carlo algorithm that samples walks along a sequence ϕ. The key idea is to consider an irreducible set of atmospheric moves (a_+, a_0, a_-) in order to construct first a digraph G as in figure 4 with vertices (states) which are self-avoiding walks and arcs which corresponds to atmospheric moves. Arcs can also be added for neutral and negative atmospheric moves; this does not change the basic operation of the algorithm. Observe that we denote by (a_+, a_0, a_-) any set of irreducible atmospheric statistics, including generalized or endpoint atmospheres.

The second step is to generate the derivative graph G' from G as explained in figure 1 and in section 2. A schematic diagram of two adjacent levels in the derivative graph is given in figure 7. Atmospheric moves (positive, neutral or negative) are illustrated by arcs from level

ℓ to level $\ell + 1$, while the bullets are the states (walks) in each level. Observe that the graph G of states is infinite, and that each level in the derivative graph is infinite while G' also have an infinite sequence of levels.

Consider a self-avoiding walk s together with its associated atmospheric statistics $a_+(s)$, $a_0(s)$ and $a_-(s)$ of positive, neutral and negative (generalized or endpoint) 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 assumed to be reversible, s is both a predecessor and a successor of s' (and vice versa), provided s' can be obtained from s through a given atmospheric move.

Let S be the state space of all self-avoiding walks from the origin. The atmospheric moves define a map $f : S \to S$ which maps the states in S to S such that $(s, s') \in f$ if s is a predecessor of s'. By starting at a state ϕ_0 and then recursively selecting elements from $f(\phi_j)$, a sequence $\phi_0, \phi_1, \phi_2, \ldots$ is realized as a random walk along the arcs in the derivative graph G', such that ϕ_j is in level j.

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 by choosing a state from the set of successors of ϕ_1. In this way, ϕ_{j+1} is selected from the successors of ϕ_j, but not necessarily with uniform probability. This process builds a sequence $\phi = \phi_0\phi_1\phi_2 \ldots \phi_j \ldots$ of states by repeated compositions of the map f defined above and in figure 7 and this composition is illustrated by the directed path in the derivate graph in figure 8.

The sampling of states from the successors of the current state is the basic operation of GAS. When the jth state is sampled, the algorithm is said to be *sample in level* j, and we illustrate this in figure 8: the algorithm starts in level zero, and then realizes a sequence $\phi = \phi_0\phi_1\phi_2 \ldots \phi_j \ldots$ by sampling successors level by level; the state ϕ_j is sampled from the jth level. Finally, the sequence ϕ is a potentially infinite directed path through the levels in the digraph in figure 8 such that state ϕ_j is in level j.

Each level S in the derivative graph in figure 8 is itself an infinite collection of states corresponding to self-avoiding walks of arbitrary length. We assume that state ϕ_j corresponds to a self-avoiding walk of length n_j in a sequence ϕ realized by GAS.

A restriction on the lengths n_j can be built into GAS as follows: {*define*} the positive atmospheres of states in S of length $n = n_{\max}$ to be equal to zero. In other words, if $\phi_j \in S$ and $n_j = n_{\max}$, then $a_+(\phi_j) = 0$.

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. With this change, atmospheric moves are still irreducible on the set of walks of length at most $n_{\max}$ and algorithm samples walks in the state space $S(n_{\max})$ which is the collection of self-avoiding walks of length at most $n_{\max}$.

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

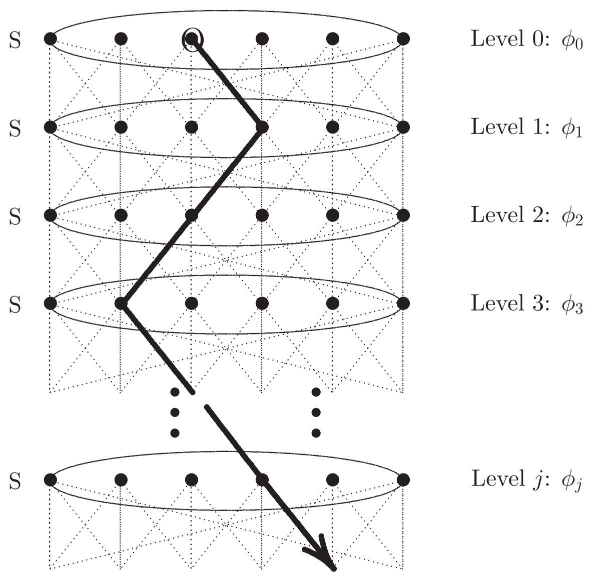

Figure 8. Sampling in the derivative graph by the GAS-algorithm. The algorithm is initiated in state ϕ_0 in level 0, marked by O. A successor state in level 1 is selected as the next state by applying an atmospheric move (positive, neutral or negative). If the current level is j, then the state ϕ_{j+1} 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 down 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 $\text{indeg}\, s = \text{outdeg}\, s = a_+^g(s) + a_0^g(s) + a_-^g(s)$.

atmospheric move on ϕ_j with probabilities

$$P_+ = P(\text{positive atmospheric move}) = \frac{\beta a_+(\phi_j)}{a_-(\phi_j) + a_0(\phi_j) + \beta a_+(\phi_j)}, \tag{9}$$

$$P_0 = P(\text{neutral atmospheric move}) = \frac{a_0(\phi_j)}{a_-(\phi_j) + a_0(\phi_j) + \beta a_+(\phi_j)}, \tag{10}$$

$$P_- = P(\text{negative atmospheric move}) = \frac{a_-(\phi_j)}{a_-(\phi_j) + a_0(\phi_j) + \beta a_+(\phi_j)}, \tag{11}$$

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 4.1 [GAS]. This algorithm samples along a sequence $\phi = \phi_0\phi_1\phi_2\ldots\phi_j\ldots$ in the state space S of self-avoiding walks where state ϕ_j is said to be in level j. If the positive atmospheres of self-avoiding walks of length $n = n_{\max}$ are defined to be zero, then the sampling is on the state space of walks of maximum length $n_{\max}$.

(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 (say $\beta \approx 1/\mu_d$), and let L be the desired length of 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 $a_+(\phi_j)$, $a_0(\phi_j)$ and $a_-(\phi_j)$.
(4) Update W_j by putting

$$W'_{j+1} = (a_-(\phi_j) + a_0(\phi_j) + \beta a_+(\phi_j))W_j.$$

(5) Compute the probabilities in equations (9)–(11). 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 on the sequence ϕ by

$$\sigma(\phi_j, \phi_{j+1}) = \begin{cases} -1, & \text{if} \quad \phi_{j+1} \text{ follows } \phi_j \text{ through } a_+, \\ +1, & \text{if} \quad \phi_{j+1} \text{ follows } \phi_j \text{ through } a_-, \\ 0, & \text{if} \quad \phi_{j+1} \text{ follows } \phi_j \text{ through } a_0. \end{cases}$$

That is, if $\phi_j \to \phi_{j+1}$ through a positive (negative) atmospheric move, then $\sigma(\phi_j, \phi_{j+1}) = -1(+1)$. Otherwise $\sigma(\phi_j, \phi_{j+1}) = 0$. 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} of weight W_{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 at step (3) to find the next state.

Define n_j to be the length (number of edges) in state ϕ_j (which is a walk of length $n_j \leqslant n_{\max}$ 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

$$W(\phi) = \left[\prod_{j=0}^{|\phi|-1}\left[\frac{a_-(\phi_j) + a_0(\phi_j) + \beta a_+(\phi_j)}{a_-(\phi_{j+1}) + a_0(\phi_{j+1}) + \beta a_+(\phi_{j+1})}\right]\right]\prod_{j=0}^{|\phi|-1}\beta^{\sigma(\phi_j,\phi_{j+1})}. \qquad (12)$$

Define $P_a(\phi)$ to be the number of positive atmospheric moves in ϕ, and $N_a(\phi)$ to be the number of negative atmospheric moves in ϕ. Then it follows that $\sum_{j=0}^{|\phi|-1}\sigma(\phi_j, \phi_{j+1}) = N_a(\phi) - P_a(\phi)$ and using this, the products above telescope down to the much simplified expression

$$W(\phi) = \left[\frac{a_-(\phi_0) + a_0(\phi_0) + \beta a_+(\phi_0)}{a_-(\phi_L) + a_0(\phi_L) + \beta a_+(\phi_L)}\right]\beta^{N_a(\phi)-P_a(\phi)}. \qquad (13)$$

This weight is very different from weights in Rosenbluth and GARM, and has the property that each time ϕ passes through the state ϕ_0, then $W(\phi) = 1$.

The probability of realizing a particular sequence ϕ is given by

$$P(\phi) = \left[\prod_{j=0}^{|\phi|-1}\left[\frac{1}{a_-(\phi_j) + a_0(\phi_j) + \beta a_+(\phi_j)}\right]\right]\beta^{P_a(\phi)} \qquad (14)$$

since the probability of particular atmospheric moves is given in equations (9)–(11).

The expected value of the weight over all sequences terminating in the state τ is then given by

$$\begin{aligned}\langle W(\phi)\rangle_\tau &= \sum_{\phi:\phi_0\to\tau} W(\phi)P(\phi) \\ &= \sum_{\phi:\phi_0\to\tau}\left[\prod_{j=0}^{|\phi|-1}\left[\frac{1}{a_-(\phi_{j+1}) + a_0(\phi_{j+1}) + \beta a_+(\phi_{j+1})}\right]\right]\beta^{N_a(\phi)}, \qquad (15)\end{aligned}$$

where the summation over $\phi : \phi_0 \to \tau$ is over all sequences starting from the state ϕ_0 and terminating in the state τ. Consider one such sequence ϕ and reverse each step in it to get the sequence ϕ'. Since the indegrees of the derivative graph are equal to its outdegrees, the atmospheres of any state in the sequence ϕ' are equal to the atmospheres of the corresponding state in the forward chain ϕ.

Each positive atmospheric step in ϕ is a negative atmospheric step in ϕ' and vice versa. Hence $N_a(\phi) = P_a(\phi')$ and we can write the summation above as

$$\langle W(\phi)\rangle_\tau = \sum_{\phi':\tau\to\phi_0} \left[\prod_{j=0}^{|\phi'|-1} \left[\frac{1}{a_-(\phi'_j) + a_0(\phi'_j) + \beta a_+(\phi'_j)} \right] \right] \beta^{P_a(\phi')}. \tag{16}$$

The summand is the probability that a sequence ϕ' starting in state τ will terminate at the state ϕ_0 if positive atmospheric moves are given with probability

$$P^+ = \frac{\beta a_+(\phi'_j)}{a_-(\phi'_j) + a_0(\phi'_j) + \beta a_+(\phi'_j)}, \tag{17}$$

neutral atmospheric moves with probability

$$P^0 = \frac{a_0(\phi'_j)}{a_-(\phi'_j) + a_0(\phi'_j) + \beta a_+(\phi'_j)}, \tag{18}$$

and negative atmospheric moves with probability

$$P^- = \frac{a_-(\phi'_j)}{a_-(\phi'_j) + a_0(\phi'_j) + \beta a_+(\phi'_j)}. \tag{19}$$

These probabilities define a backwards Markov chain starting in the state τ and terminating in the state ϕ_0. If the set of atmospheric moves is irreducible, and if the mean probabilities of the (backwards) positive and negative atmospheric moves P^+ and P^- in the backwards chain given by equations (17) and (19) satisfy

$$\langle P^+\rangle_n \leqslant \langle P^-\rangle_n \tag{20}$$

over the entire range of lengths of walks in the backwards chain, then the probability in equation (16) (that the backwards chain terminates in the state ϕ_0) is bigger than zero, since the atmospheric moves are reversible and the entire process is a (biased) random walk on the integers with average transition probabilities $\langle P^-\rangle_n$ for a transition $n \to n-1$ and $\langle P^+\rangle_n$ for transition $n \to n+1$. Since the process is ergodic (aperiodic and irreducible), the state $n = 0$ is persistent and the process terminates with positive probability at $n = 0$ in the state ϕ_0.

This is in particular true if β in equation (19) satisfies

$$\beta \leqslant \frac{\langle a_-\rangle_n}{\langle a_+\rangle_n} \tag{21}$$

for all values of n, or whenever

$$\beta \leqslant \inf_n \left[\frac{\langle a_-\rangle_n}{\langle a_+\rangle_n} \right]. \tag{22}$$

If the algorithm is defined with a maximum value on the length of the states, say $n_{\max}$, then the restriction is not needed: the probabilities above define an ergodic Markov process on $[0, n_{\max}]$ and the state ϕ_0 is persistent. Thus, for any finite value of β, the mean weight in equations (15) and (16) is positive.

The average weight $\langle W(\phi)\rangle$ is asymptotically independent of the starting state τ in equation (16) in the backwards Markov process:

$$\langle W(\phi)\rangle_\tau = \sum_{\phi':\tau\to\phi_0} P(\phi_0|\tau) \to C_0 > 0, \tag{23}$$

where $P(\phi_0|\tau)$ is the conditional probability that a sequence will terminate in ϕ_0 given that it started in τ and where C_0 is asymptotically independent of τ (as $|\phi'| \to \infty$). Summing this over all sequences of length $L = |\phi|$ levels (with L large) ending in states τ of length n shows that

$$\sum_{|\tau|=n} \langle W(\phi)\rangle_\tau \to c_n \langle P(\phi_0|\tau)\rangle_n, \tag{24}$$

where c_n is the number of walks of length n, and $\langle P(\phi_0|\tau)\rangle_n$ is the mean conditional probability that the sequence terminates in ϕ having started in τ. Similarly, for walks σ of length m it follows that

$$\sum_{|\sigma|=m} \langle W(\phi)\rangle_\sigma \to c_m \langle P(\phi_0|\sigma)\rangle_m. \tag{25}$$

Asymptotically, the averages $\langle P(\phi_0|\tau)\rangle_n$ are independent of n, and these factors cancel when the ratio of equations (24) and (25) is taken. This gives

$$\frac{\sum_{|\tau|=n} \langle W(\phi)\rangle_\tau}{\sum_{|\sigma|=m} \langle W(\phi)\rangle_\sigma} \to \frac{c_n}{c_m} \qquad \text{as} \quad |\phi| \to \infty. \tag{26}$$

In particular, if $m = 0$, then $c_0 = 1$ and

$$\frac{\sum_{|\tau|=n} \langle W(\phi)\rangle_\tau}{N_0} \to c_n \qquad \text{as} \quad |\phi| \to \infty, \tag{27}$$

since the weight of a sequence terminating in the state ϕ_0 is 1, and where N_0 is the number of visits of the sequence ϕ to ϕ_0. This last expression is the ratio of the accumulated weight of states of length n along the sequence ϕ, to the accumulated weight of ϕ_0.

In practical implementations, the total (unnormalized) accumulated weight $\sum_{|\tau|=j} \langle W(\phi)\rangle_\tau$ is collected along sequences ϕ (of length L states, for L large) realized by the algorithm. Data are collected for a state τ each time the chain passes through τ, and ratios of the accumulated weights produce estimates of ratios of c_n as in equations (26) and (27). Normally, a simulation will proceed by generating a sequence ϕ with L levels, and estimating accumulated weights by binning the weights for each value of n. This gives the accumulated weights in equation (26) from which c_n can be estimated.

The implementation of GAS proceeds by realizing N independent sequences ϕ and computing weights for each. This provides one with independent estimates for c_n/c_m, which can then be analysed. 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.

In figure 9, the distribution of walks sampled along a sequence ϕ of length $L = 50 \times 10^6$ levels is plotted against length n in a simulation of GAS with $\beta = 0.225$ in two dimensions using generalized atmospheres. The histogram shows that short walks were sampled often, and that the sampling of longer walks decreases exponentially with n (this is to be expected, since $\beta < 1/\mu$ and $\left[\left\langle a_-^g\right\rangle_n / \left\langle a_+^g\right\rangle_n\right] = c_n/c_{n+1}$—generally it is thought that $[c_{n+1}/c_n] \to \mu$). In general, it turns out to be tricky to tune β to obtain good sampling over a range of values of n, although the imposition of a maximum value of n improves the situation somewhat.

Estimates of c_n obtained from the data in figure 9 are listed in table 1 by normalizing accumulated weights with respect to the weight of the trivial walk. We observe that for small n the algorithm produces good estimates, but that the accuracy decreases quickly with increasing n, partly because fewer walks were realized along the sequence for larger values of n. In the following section, we illustrate a method for implementing GAS such that it self-tunes to produce flat-histogram sampling and to improve the estimates of c_n for larger values of n.

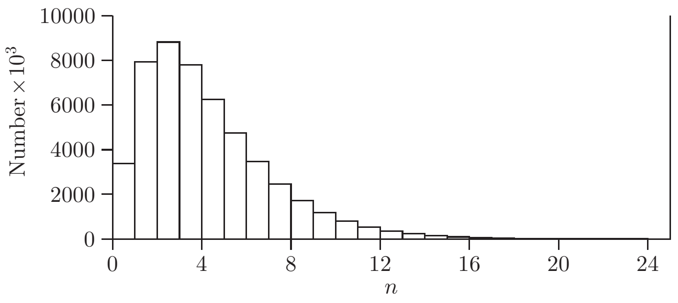

Figure 9. Sampling by the GAS-algorithm. In this run of two-dimensional walks, we put $\beta = 0.225$ and sampled a (single) sequences of length $L = 50 \times 10^6$ levels. The histogram above gives the number of states binned according to length n. Since $\beta < 1/\mu$ in this example, longer walks were sampled less often, and the attrition is exponential with increasing n. By determining the average weights of states of length n as in equation (24) and taking the ratios of average weights (see equation (27), estimates of c_n can be determined. The results are displayed in table 1. Since the sampling is poor for longer walks, the estimate of c_n deteriorates quickly with increasing n.

Table 1. Approximate enumeration with GAS.

n	c_n in 2D	Estimate
0	1	1
1	4	4.001 43
2	12	12.0024
3	36	35.9801
4	100	99.9584
5	284	283.824
6	780	778.920
7	2172	2169.58
8	5916	5916.18
9	16 268	16 282.3
10	44 100	44 158.9
11	120 292	120 566
12	324 932	325 657
13	881 500	882 577
14	2374 444	2377 630
15	6416 596	6371 060
16	17 245 332	17 021 200

The advantage of flat-histogram sampling of GAS over other flat-histogram methods such as flatPERM [22] and GARM [24], is that the flat histogram is obtained by tuning β, rather than by using enrichment and pruning techniques. The implementation is therefore simpler, and there should be fewer correlations between states sampled along a sequence ϕ.

5. Flat histogram generalized atmospheric sampling

In this section, we describe an algorithm for implementing a flat-histogram version of GAS. We modify the parameter β in algorithm 4.1 such that states of length n are sampled from a flat histogram over $[0, n_{\max}]$, except at the endpoints of this interval. As before, we denote by (a_+, a_0, a_-) any set of irreducible atmospheric statistics, including generalized or endpoint atmospheres.

Define the state space $S(n_{\max})$ to be all the walks from the origin of lengths in $[0, n_{\max}]$. An implementation of GAS on $S(n_{\max})$ does not give a flat histogram on $[0, n_{\max}]$, but operates by sampling states ϕ_j of lengths $n_j \in [0, n_{\max}]$ such that the sequence $\langle n_j \rangle$ is a (biased) random walk on the integers in $[0, n_{\max}]$. Since the algorithm is irreducible, it will sample all walks of lengths $n \in [0, n_{\max}]$ with positive probability. The aim is to sample uniformly in $[0, n_{\max}]$.

We proceed by making the parameter β in the probabilities in equations (9)–(11) dependent on the length n_j of the state ϕ_i along the sequence ϕ. In particular, we choose β such that

$$\langle P^+\rangle_n = \langle P^-\rangle_n \tag{28}$$

in equations (17) and (19); this makes a positive atmospheric step as likely as a negative atmospheric step on average for each value of n in $(0, n_{\max})$. By replacing β by β_n in equations (9)–(11) with

$$\beta_n = \frac{\langle a_-\rangle_n}{\langle a_+\rangle_n}, \tag{29}$$

GAS will select a longer walk as the next state with average probability equal 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 5.1 [flat-histogram GAS]. This algorithm samples along a sequence $\phi = \phi_0\phi_1\phi_2\ldots\phi_j\ldots$ in the state space $S(n_{\max})$ of walks where state ϕ_j is said to be in level j and has length n_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_n at a convenient value for each $n \in [0, n_{\max}]$, and let L be the desired length (number of levels) 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 $a_+(\phi_j)$, $a_0(\phi_j)$ and $a_-(\phi_j)$. Note that $a_+(\phi_j) = 0$ if $n_j = n_{\max}$.

(4) Update W_j by putting

$$W'_{j+1} = (a_-(\phi_j) + a_0(\phi_j) + \beta_{n_j} a_+(\phi_j))W_j.$$

(5) Compute the probabilities

$$P_+ = P(\text{positive atmospheric move}) = \frac{\beta_{n_j} a_+(\phi_j)}{a_-(\phi_j) + a_0(\phi_j) + \beta_{n_j} a_+(\phi_j)}, \tag{30}$$

$$P_0 = P(\text{neutral atmospheric move}) = \frac{a_0(\phi_j)}{a_-(\phi_j) + a_0(\phi_j) + \beta_{n_j} a_+(\phi_j)}, \tag{31}$$

$$P_- = P(\text{negative atmospheric move}) = \frac{a_-(\phi_j)}{a_-(\phi_j) + a_0(\phi_j) + \beta_{n_j} a_+(\phi_j)}. \tag{32}$$

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 on the sequence ϕ by

$$\sigma(\phi_j, \phi_{j+1}) = \begin{cases} -1, & \text{if} \quad \phi_{j+1} \text{ follows } \phi_j \text{ through } a_+, \\ +1, & \text{if} \quad \phi_{j+1} \text{ follows } \phi_j \text{ through } a_-, \\ 0, & \text{if} \quad \phi_{j+1} \text{ follows } \phi_j \text{ through } a_0. \end{cases}$$

That is, if $\phi_j \to \phi_{j+1}$ through a positive (negative) atmospheric move, then $\sigma(\phi_j, \phi_{j+1}) = -1(+1)$. Otherwise $\sigma(\phi_j, \phi_{j+1}) = 0$. Update the weight by

$$W_{j+1} = \frac{W'_{j+1} \beta_{n_{j+1}}^{\sigma(\phi_j,\phi_{j+1})}}{(a_-(\phi_{j+1}) + a_0(\phi_{j+1}) + \beta_{n_{j+1}} a_+(\phi_{j+1}))}.$$

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 at step (3) to find the next state. If the sequence was terminated, then update estimates for β_n for each $n \in [0, n_{\max}]$ by computing

$$\beta_n = \frac{\langle a_- \rangle_n}{\langle a_+ \rangle_n}.$$

Since $a_-(s) = 0$ if s is the walk of zero length, put $\beta_0 = 1$. Since $a_+(s) = 0$ if s is a walk of length $n_{\max}$, put $\beta_{n_{\max}} = 0$. With this new set of β_n, start a new sequence from step (1). Repeat this until a desired number N of sequences have been realized.

(8) If each of the sequences ϕ is sufficiently long (L is large), then each new sequence will generate a flatter histogram over the lengths $n \in [0, n_{\max}]$.

This implementation of GAS proceeds by realizing N independent sequences ϕ of L levels each and computing weights for each while updating the values of β_n following the completion of each sequence. If the initial choices for β_n 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 half as often on average (GAS performs a random walk on the integers in $[0, n_{\max}]$, with reflecting boundaries).

The same arguments developed for GAS following algorithm 4.1 shows that the weight of a sequence ϕ is given by

$$W(\phi) = \left[\frac{a_-(\phi_0) + a_0(\phi_0) + \beta_{n_0} a_+(\phi_0)}{a_-(\phi_L) + a_0(\phi_L) + \beta_{n_L} a_+(\phi_L)}\right] \prod_{j=0}^{|\phi|-1} \beta_{n_j}^{\sigma(\phi_j,\phi_{j+1})} \tag{33}$$

and the average weights of sequences can be used to estimate c_n by computing the ratio

$$R_{n,m} = \frac{\sum_{|\tau|=n} \langle W(\phi) \rangle_\tau}{\sum_{|\sigma|=m} \langle W(\phi) \rangle_\sigma} \to \frac{c_n}{c_m} \qquad \text{as} \quad |\phi| \to \infty. \tag{34}$$

In figure 10, an example of flat-histogram GAS sampling of walks using generalized atmospheres is given. The length of the sequence was $L = 1000\,000$ levels, and values of β_n were computed from an earlier run using equation (29); these are denoted by the solid bars. The binning of states of length n in figure 10 shows that GAS effectively performs a random walk on $n \in [0, n_{\max}]$, producing the nearly flat histogram indicated by the open bars.

In general, it was easy to estimate values of β_ℓ to give flat distributions, even over wide intervals with large values of $n_{\max}$. Generally we proceeded by computing weights along each sequence, while β_ℓ were updated after each sequence to initiate the next. This produced flat distributions on $[0, n_{\max}]$ similar to the data displayed in figure 10. Generally, updating values of β_ℓ to obtain a flat distribution was efficient: in figure 11, we display the initial values of β_ℓ

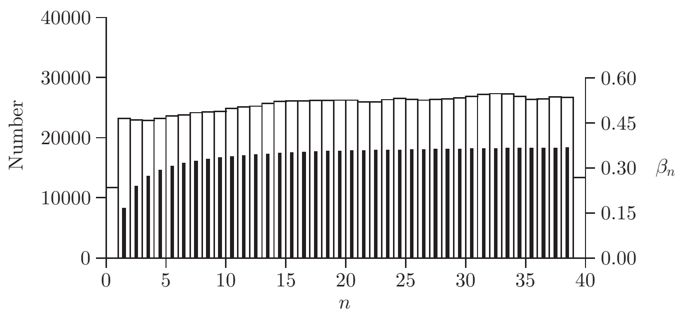

Figure 10. Flat-histogram sampling in the GAS-algorithm. The solid bars give the values of β_n as a function of n on the right-hand side scale. The bar corresponding to $\beta_0 = 1$ is left away, since the probability of stepping from the walk of length 0 to a walk of length 1 is $1/2d$, independent of β_0. In this simulation, $n_{\max} = 40$, and so $\beta_{40} = 0$. The values of β_n were computed by atmospheric ratios (see equation (29)) from an earlier run. The binning of states of length $n \in [0, 40]$ in a sequence of length 1000 000 with β_n as given, is indicated by the open bars with scale on the left-hand axis. The states with $n = 0$ and $n = 40$ were sampled about half as frequently as those states with $n \in (0, 40)$.

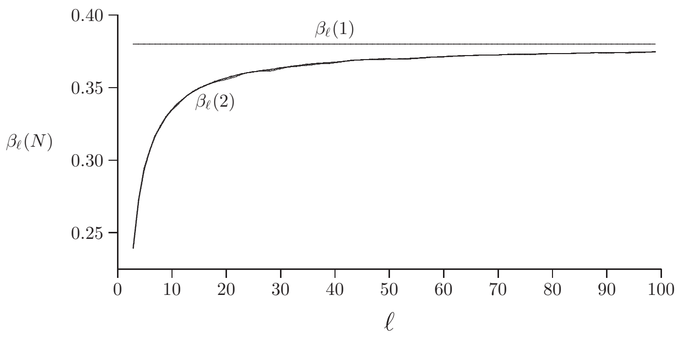

Figure 11. Determining β_ℓ for a GAS simulation. A good initial choice for β_ℓ is $\beta_\ell(1) = 1/\mu$ for all ℓ in $[0, n_{\max}]$ (this is represented by the horizontal line). After each GAS-sequence, β_ℓ is updated as in equation (29) to obtain $\beta_\ell(N)$ for $N = 2, 3, 4, \ldots$. The updated values of $\beta_\ell(N)$ are displayed for sequences $N = 2, 3, 4$ along the curve denoted by $\beta_\ell(2)$. These values of β_ℓ are virtually unchanged for all subsequent sequences and have stabilized to give flat sampling after one initial sequence was generated. In this simulation $n_{\max} = 100$ and each sequence consists of 10^6 levels.

for a simulation on [0, 100] in $d = 2$ with just four sequences of length $L = 10^6$. The initial choice was $\beta_\ell = 0.38 \approx 1/\mu$. The updated values of β_ℓ stabilized after just one sequence was used to update β_ℓ, and all further updates were of no consequence in the simulations. The data for the second, third and fourth sequences all collapse to the single curve in figure 11. This kind of behaviour was observed in all our simulations, and we noted that a good initial choice for β_ℓ is $\beta_\ell \approx 1/\mu_d$.

6. Numerical results

Versions of GAS were coded using either positive and negative endpoint atmospheric moves (see figure 2), or positive and negative generalized atmospheric moves (see figures 5 and 6).

We call the endpoint atmospheric implementation 'GABS', because endpoint atmospheric moves were used in the Beretti–Sokal algorithm [3].

The implementation of GABS relies superficially on the same elementary moves as the Beretti–Sokal algorithm, but the dynamics of the algorithm are completely different. The basic advantage of the endpoint atmospheric implementation of GABS is that an elementary move can be performed in $O(1)$ CPU-time. This implies that very long weighted sequences ϕ can be generated, and moreover, that the total CPU-time is insensitive of $n_{\max}$, the maximum length of self-avoiding walks sampled by the algorithm.

In contrast to this, the use of generalized atmospheres in GAS requires more CPU-time. An average generalized atmospheric elementary move requires $O(n)$ CPU-time, on a self-avoiding walk of length n. Moreover, since the distribution is flat over walks of lengths $n \in (0, n_{\max})$, the CPU-time per elementary move increases on average linearly with $n_{\max}$. This dependence may be improved with better use of data structures (see for example [4]), but we did not pursue that in this study.

We performed several runs for different values of $n_{\max}$. Our purpose was to compute average weights (see equation (34)) in order to estimate c_n. By computing the ratio $R_{n,m}$ in equation (34) for several values of m, one may estimate c_n by

$$c_n \approx \frac{1}{N+1} \sum_{m=0}^{N} c_m R_{n,m}. \tag{35}$$

Since c_m is known for small values of m, this approximation estimates c_n in terms of c_m for small values of m. In our initial simulations, we chose $N = 2$ in the above, and then compared estimates for c_n with values obtained by exact enumeration studies [12, 15].

6.1. Results from simulations using GAS

Runs with $n_{\max} = 249$ in the two- and three-dimensional hypercubic lattices were performed. In two dimensions, GAS realized 200 sequences ϕ, each sequence with 50×10^6 levels. In three dimensions, GAS was used to sample along 300 sequences ϕ, each sequence with 50×10^6 levels.

Average weights were computed and c_n were estimated using equation (35) with $N = 2$. In table 2, we compare estimates for c_n extracted from our simulations with exact results.

The results in table 2 and estimates for c_n for $n \in [0, 249]$ suggest that one may compute an estimate for the growth constant μ and entropic exponent γ of walks in two and three dimensions. Since our data are essentially inexact series, one should be able to use series analysis techniques, but we have only been able to use the most basic methods; our data were not smooth enough for more sophisticated approaches.

The ratio estimator

$$r_n = \sqrt{\frac{c_{n+2}}{c_n}} \sim \left(1 + \frac{\gamma - 1}{n}\right) \mu \tag{36}$$

can be plotted against $1/n$ and by extrapolating it to its intercept with the Y-axis an estimate of μ is obtained. By examining the dispersion of the ratio estimators and extrapolating it to large n, one can bound the growth constant of walks: $2.6368 \leqslant \mu \leqslant 2.6388$. We plot these ratio estimators $n \in [100, 249]$ in figure 12.

A better estimate is obtained if the linear extrapolant

$$\mu_n = n r_n - (n-1) r_{n-1} \tag{37}$$

is instead computed. Assuming that μ_n are normally distributed about μ for $n \geqslant n_0$, and by taking an average, one obtains estimates of μ virtually independent of n_0

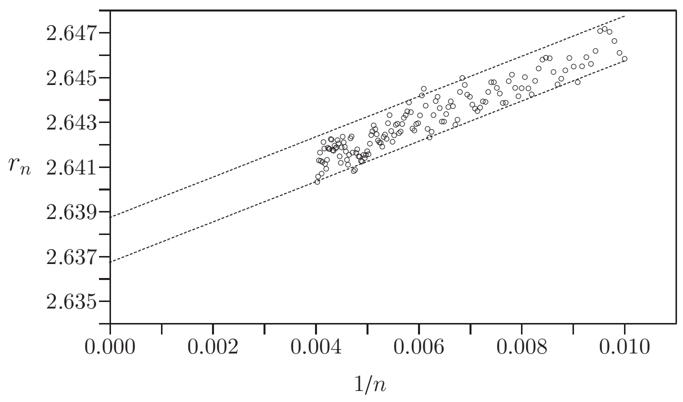

Figure 12. The ratio estimator for data produced by GAS in two dimensions is plotted against $1/n$ for $n \in [100, 249]$.

Table 2. Approximate enumeration with GAS.

n	c_n in 2D	Estimate	c_n in 3D	Estimate
0	1	1.000 25	1	0.999 616
1	4	4.000 42	6	5.998 52
2	12	11.9994	30	30.0018
3	36	36.0038	150	150.039
4	100	100.030	726	726.132
5	284	284.153	3534	3535.13
6	780	780.485	16 926	16 939.0
7	2172	2173.40	81 390	81 489.0
8	5916	5920.84	38 766	388 482
9	16 268	16 276.6	1853 886	1856 250
10	44 100	44 110.9	8809 878	8818 800
11	120 292	120 318	41 934 150	41 964 600
12	324 932	324 928	198 842 742	198 945 000
13	881 500	881 350	943 974 510	944 280 700
14	2374 444	2374 275	4468 911 678	4470 190 000
15	6416 596	6418 170	21 175 146 054	21 178 000 000
16	17 245 332	17 253 000	100 121 875 974	100 110 400 000

for small values of n_0. If we denote these estimates in the format (n_0, μ_{n_0}), then our results are $(1, 2.638\,30\ldots)$, $(5, 2.638\,32\ldots)$, $(10, 2.638\,31\ldots)$, $(15, 2.638\,00\ldots)$ and $(20, 2.638\,06\ldots)$. Considering the spread in these results, the midpoint is around $\mu = 2.6382$, with a confidence interval which we take to be one-half the difference in the spread. This gives

$$\mu = 2.6382 \pm 0.0002 \tag{38}$$

as a best estimate for μ from these extrapolants.

Similarly, biased estimates for γ are given by

$$\gamma_n = nr_n/\mu - n + 1, \tag{39}$$

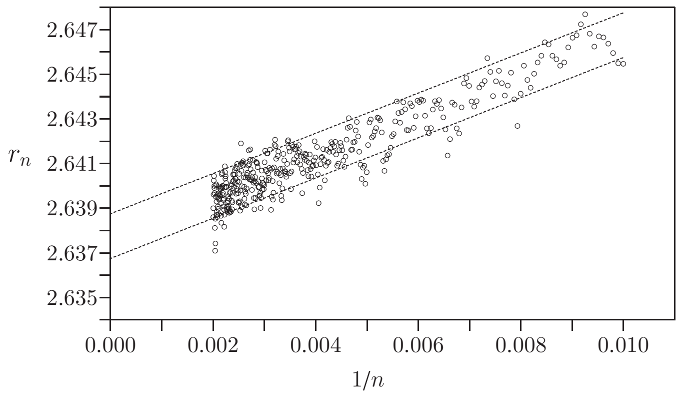

Figure 13. A magnification of the *Y*-intercept of data obtained by implementing GAS for $n \in [0, 499]$. In this plot, r_n is displayed against $1/n$ for $n \in [100, 499]$. These data indicate that $2.6368 \leqslant \mu \leqslant 2.6388$.

where we may choose $\mu = 2.6382$ from the results above. Assuming that $\mu = 2.6382$ and taking the average for $n \geqslant n_0$ give results virtually independent of n_0 for $n_0 > 1$. By plotting γ_n against $1/n$, one obtains the estimate $1.32 \leqslant \gamma \leqslant 1.36$. By taking the midpoint as our best estimate, and taking one half of the difference of the bounds as a confidence interval, our best estimate is

$$\gamma = 1.34 \pm 0.02. \tag{40}$$

These results for μ and γ are close to the exact enumeration data value for μ given by $\mu = 2.638\,15\ldots$ ([15], see also references [12, 23]) and the exact value of the entropic exponent: $\gamma = 43/32 = 1.343\,75$ [7, 21].

We have also examined our data by other series techniques such as Padè and differential approximants, but the approximate series produced by GAS are still too noisy to give consistent results. That is, while some approximants give results which are good estimates for μ, other nearby approximants did not converge at all, indicating that the methods are numerically unstable or unreliable.

To gauge the accuracy of the simulations, we repeated the GAS calculation in two dimensions for $n \in [0, 499]$, sampling along 200 sequences, each sequence of length 50×10^6. The results were similar to the above. In figure 13, the ratio estimator r_n is plotted against $1/n$, magnified and displayed for $n \in [100, 499]$. The results are consistent with those obtained in the interval $[0, 249]$, confirming the analysis and giving a consistent estimate of the confidence interval claimed above for the extrapolated value of μ: $2.6368 \leqslant \mu \leqslant 2.6388$. Similar extrapolation of γ_n in equation (39) gives $\gamma \approx 1.33$, close to its exact value.

In three dimensions the ratio estimator in equation (36) was first computed from $n \in [100, 249]$. By plotting it against $1/n$ and magnifying and extrapolating the data to large values of n, this shows that $4.6825 \leqslant \mu \leqslant 4.6860$ in the cubic lattice. See figure 14 for a plot of r_n against $1/n$ for $n \in [100, 249]$.

Linear extrapolation through equation (37) gives linear extrapolants μ_n. Assuming that μ_n are normally distributed about μ and by minimizing the least squares error, and giving our results in the form (n_0, μ_{n_0}), we obtain $(1, 4.6820\ldots)$, $(5, 4.6835\ldots)$, $(10, 4.6845\ldots)$, $(15, 4.6853\ldots)$ and $(20, 4.6842\ldots)$. If one ignores the apparent outlier at $n_0 = 1$, then the

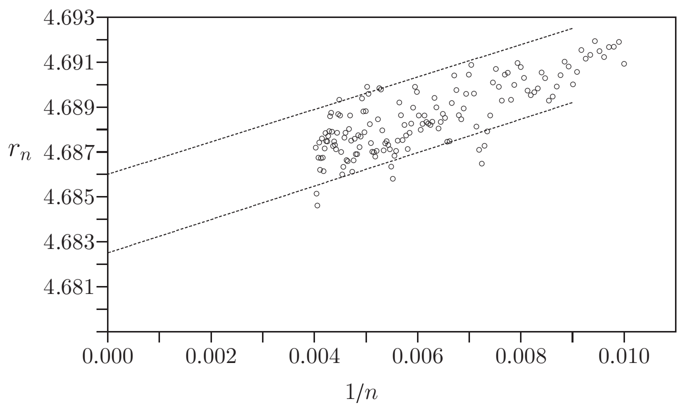

Figure 14. The ratio estimator for data produced by GAS in three dimensions is plotted against $1/n$ for $n \in [100, 249]$.

spread of estimates have midpoint 4.6844 . . . and if one-half the spread is taken as a confidence interval, then

$$\mu = 4.6844 \pm 0.0009 \tag{41}$$

is our best estimate from these data.

Biased estimates for γ are given by the extrapolant in equation (39) where we choose μ equal to its average $\mu = 4.6844$ obtained above. By plotting γ_n against $1/n$, one obtains the estimate $1.14 \leqslant \gamma \leqslant 1.18$. By taking the midpoint as our best estimate, and taking one half of the difference of the bounds as a confidence interval, our best estimate is

$$\gamma = 1.16 \pm 0.02. \tag{42}$$

This result is consistent with estimates of the entropic exponent obtained elsewhere ($\gamma = 1.160 \pm 0.004$ [16–18]).

6.2. Results from simulations using GABS

Runs with $n_{\text{max}} = 999$ in the two- and three-dimensional hypercubic lattices were performed. Since the implementation of endpoint atmospheric moves in this algorithm is fast, we were able to realize 3000 sequences ϕ, each sequence with 500×10^6 levels in both two and three dimensions.

Weights were computed along each sequence, while β_n were updated after each sequence to initiate the next. This produced flat distributions on [0, 999] similar to the data displayed in figure 10.

Average weights were computed and c_n were estimated using equation (35) with $N = 2$. In table 3, we compare estimates for c_n extracted from our simulations with exact results.

In two dimensions the ratio estimator r_n in equation (36) is plotted against $1/n$ in figure 15 for $n \in [100, 999]$. By extrapolating these data to the Y-axis one obtains an estimate of μ. By magnifying and extrapolating the data as shown, it follows that $2.6378 \leqslant \mu \leqslant 2.6384$ in two dimensions.

A second estimate can be obtained by computing the linear extrapolant μ_n in equation (37). The results are displayed against $1/n_0$ in figure 16, and the extrapolants accumulate close to 2.638. Analysing these data as before gives estimates of μ virtually independent of

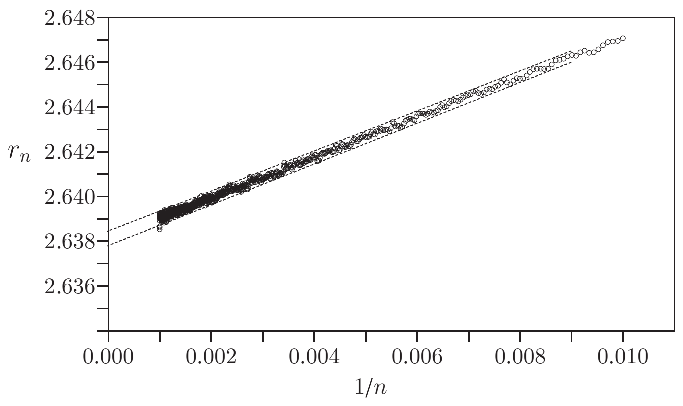

Figure 15. The ratio estimator r_n (see equation (36)) plotted against $1/n$ for two-dimensional data obtained by GABS for $n \in [2, 999]$. Displayed are data for values of $n \in [100, 999]$. Extrapolating to 0 gives an estimate of μ.

Table 3. Approximate enumeration with GABS.

n	c_n in 2D	Estimate	c_n in 3D	Estimate
0	1	0.999 96	1	1.000 07
1	4	3.999 87	6	6.000 17
2	12	12.000 17	30	29.999 77
3	36	36.003 57	150	150.0053
4	100	100.0210	726	726.0873
5	284	284.0680	3534	3534.620
6	780	780.2060	16 926	16 929.37
7	2172	2172.580	81 390	81 407.93
8	5916	5917.757	387 966	388 041.0
9	16 268	16 273.50	1853 886	1854 240
10	44 100	44 117.10	8809 878	8811 953
11	120 292	120 343.7	41 934 150	41 944 630
12	324 932	325 092.7	198 842 742	198 897 700
13	881 500	881 974.7	943 974 510	944 239 300
14	2374 444	2375 687	4468 911 678	4470 120 000
15	6416 596	6419 573	21 175 146 054	21 180 730 000
16	17 245 332	17 253 130	100 121 875 974	100 147 670 000

n_0 for small values of n_0. In particular, if we present our results in the form (n_0, μ_{n_0}), then we obtain $(1, 2.637\,83\ldots)$, $(5, 2.638\,31\ldots)$, $(10, 2.638\,37\ldots)$, $(15, 2.638\,31\ldots)$ and $(20, 2.638\,23)$. Taking the midpoint of the spread as our best estimate, and half the spread as a confidence interval, gives, while ignoring the apparent outlier at $n_0 = 1$, our best estimate

$$\mu = 2.638\,30 \pm 0.000\,07. \tag{43}$$

Estimating the entropic exponent from our data proved more difficult. The extrapolant γ_n (see equation (39), with $\mu = 2.6383$) is plotted against $1/n$ in figure 17. There is a linear relationship for smaller values of n, but numerical noise in the data accumulates with increasing n to produce a large spread as n approaches 999. Least squares analysis of the data

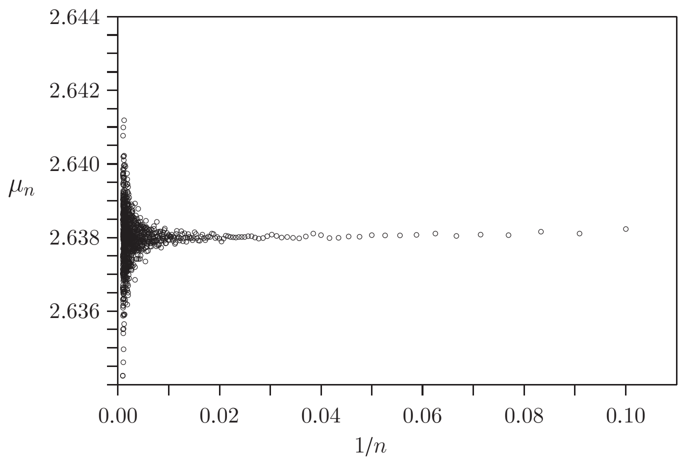

Figure 16. Linear extrapolants μ_n (see equation (37)) plotted against $1/n$ for two-dimensional data obtained by GABS for $n \in [2, 999]$. Displayed are data for values of $n \in [10, 999]$. Extrapolating gives an estimate of μ.

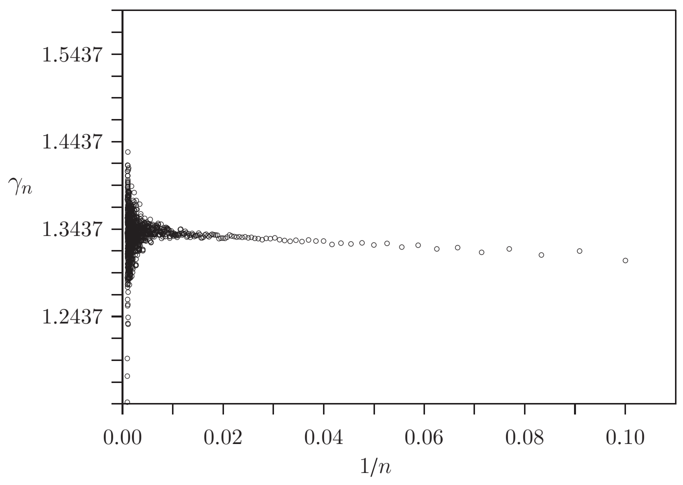

Figure 17. Linear extrapolants γ_n (see equation (39)) plotted against $1/n$ for two-dimensional data obtained by GABS for $n \in [2, 999]$. Displayed are data for values of $n \in [10, 999]$.

produced values of γ too small if all the data are included, but by drawing bounds on the data for small values of n, one may conclude that $1.32 \leqslant \gamma \leqslant 1.36$. This shows that

$$\gamma = 1.34 \pm 0.02. \tag{44}$$

This is consistent with the estimate obtained in two dimensions from GAS data in $[0, 249]$.

In three dimensions, the ratio estimator r_n in equation (36) is plotted against $1/n$ in figure 18 for $n \in [100, 999]$. By extrapolating these data to the Y-axis one obtains bounds on an estimate of μ. By magnifying and extrapolating the data as shown in the figure, it follows that $4.6835 < \mu < 4.6845$ in three dimensions.

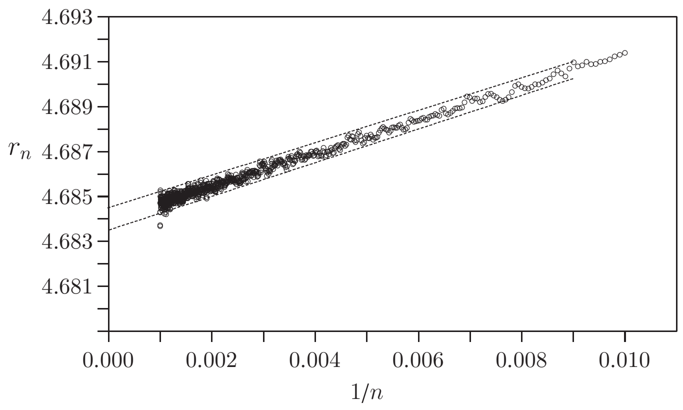

Figure 18. The ratio estimator r_n (see equation (36)) plotted against $1/n$ for three-dimensional data obtained by GABS for $n \in [2, 999]$. Displayed are data for values of $n \in [100, 999]$. Extrapolating to 0 gives an estimate of μ.

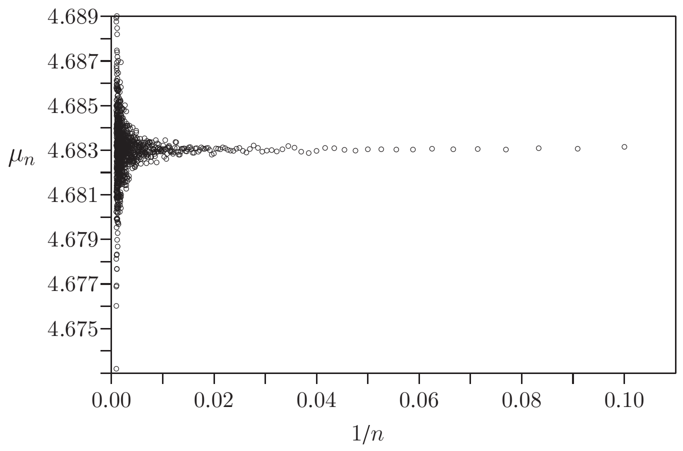

Figure 19. Linear extrapolants μ_n (see equation (37)) plotted against $1/n$ for three-dimensional data obtained by GABS for $n \in [2, 999]$. Displayed are data for values of $n \in [10, 999]$.

A second estimate can be obtained by computing the linear extrapolant μ_n in equation (37). The results are displayed against $1/n_0$ in figure 19, and the extrapolants accumulate close to 4.684. Analysing these results gives estimates of μ virtually independent of n_0 for small values of n_0. In particular, if we present our results in the form (n_0, μ_{n_0}), then we obtain $(1, 4.6830\ldots)$, $(5, 4.6844\ldots)$, $(10, 4.6841\ldots)$, $(15, 4.6840\ldots)$ and $(20, 4.6838)$. Taking the midpoint of the spread as our best estimate, and half the spread as a confidence interval gives, while ignoring the apparent outlier at $n_0 = 1$, we obtain our best estimate

$$\mu = 4.6837 \pm 0.0007. \tag{45}$$

Similarly, biased estimates for γ are given by the estimator in equation (39). Assuming that $\mu = 4.6837$ and plotting the data gives the results in figure 20. There is a linear relationship for smaller values of n, but numerical noise in the data accumulates with increasing n to produce

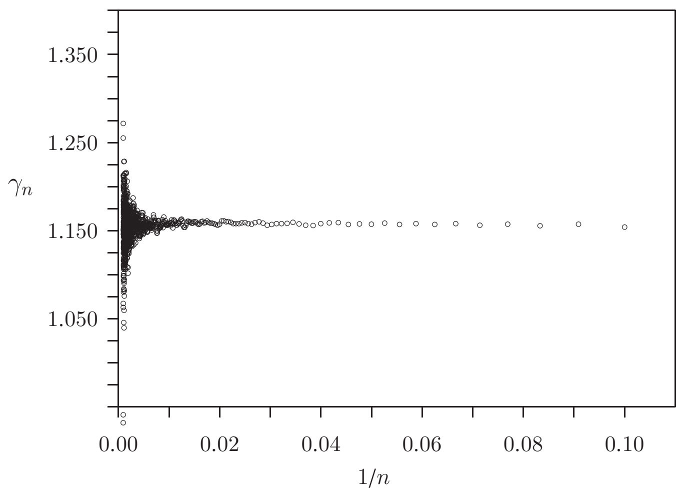

Figure 20. Linear extrapolants γ_n (see equation (39)) plotted against $1/n$ for three-dimensional data obtained by GABS for $n \in [2, 999]$. Displayed are data for values of $n \in [10, 999]$.

a large spread as n approaches 999. Least squares analysis of the data produced values of γ too small if all the data are included, but by drawing bounds on the data for small values of n, one may conclude that $1.140 \leqslant \gamma \leqslant 1.165$. This shows that

$$\gamma = 1.153 \pm 0.013. \tag{46}$$

This is consistent with the estimate obtained in three dimensions from GAS data in [0, 249].

7. Conclusions

In this paper, we proposed a new Monte Carlo algorithm for sampling self-avoiding walks. The algorithm is a generalization of GARM [24], and it naturally provides a method for sampling from flat histograms without invoking pruning and enrichment techniques as in PERM [11]. The algorithm is very general in the sense that any selection of irreducible atmospheric moves can be used, and we considered only two cases: a generalized atmosphere and an endpoint atmosphere.

We used the algorithm to sample self-avoiding walks using generalized and endpoint atmospheres, and showed that the algorithm is an approximate enumeration technique. We analysed our results by using basic series analysis techniques (linear extrapolants) to estimate the growth constant and entropic exponents for self-avoiding walks. The estimates for μ and γ can be rounded by considering in each case the confidence interval. In this event, we state our best values for μ and γ from the simulations as follows.

The estimates for μ and γ from GAS simulations for $n \in [0, 249]$ give the estimates

$$\mu = 2.6382 \pm 0.0002 \qquad \text{and} \qquad \gamma = 1.34 \pm 0.02 \tag{47}$$

in two dimensions, and

$$\mu = 4.684 \pm 0.001 \qquad \text{and} \qquad \gamma = 1.16 \pm 0.02 \tag{48}$$

rounded up to the next digit, in three dimensions.

Simulations using the GABS version of GAS are more efficient because the implementation of endpoint atmospheric moves is computationally fast. This enabled us

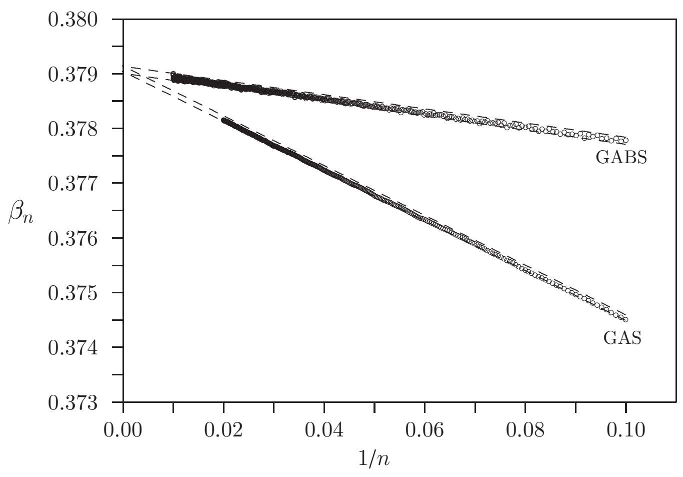

Figure 21. Extrapolating $\beta_n = [\langle a_- \rangle_n / \langle a_+ \rangle_n]$ to $n \to \infty$ for generalized atmospheres (GAS) and endpoint atmospheres (GABS) in two-dimensional simulations of GAS and GABS. Since β_n should approach $1/\mu$ as $n \to \infty$, the Y-intercepts are estimates of $1/\mu$.

to sample walks of lengths in the interval [0, 999] in two and three dimensions. Analysing the results and considering the confidence intervals in each case give our best estimates as follows:

$$\mu = 2.6383 \pm 0.0001 \qquad \text{and} \qquad \gamma = 1.34 \pm 0.02 \tag{49}$$

in two dimensions, and

$$\mu = 4.684 \pm 0.001 \qquad \text{and} \qquad \gamma = 1.15 \pm 0.02 \tag{50}$$

in three dimensions. The estimates of μ in this case are consistent with the GAS estimates in equations (47) and (48).

Since the GABS estimates involve sampling over much longer sequences and lengths of walks, we take the estimates in equations (49) and (50) as our best estimates for μ, and note that these estimates are within the error bars or identical to the results in equations (47) and (48).

The self-tuning of β_n in GAS and GABS allows us to extrapolate β_n to β_∞ as $n \to \infty$. We do this in figure 21 for GAS and GABS in two dimensions. Carefully extrapolating the data as shown, shows that $0.3789 \leqslant \beta_\infty \leqslant 0.3791$ (GAS data) and $0.3789 \leqslant \beta_\infty \leqslant 0.3791$ (GABS data). Since $\beta_n \to 1/\mu$, these numerical bounds give estimates for the growth constant, and we obtain $\mu = 2.6385 \pm 0.0007$ from both data sets, which we round up to

$$\mu = 2.6385 \pm 0.0010 \tag{51}$$

as a best value. This compares well with our best estimate in equation (49) above.

The same arguments can be used in the three-dimensional data, and the extrapolation of β_n is illustrated in figure 22. Carefully extrapolating the data as shown, shows that $0.21345 \leqslant \beta_\infty \leqslant 0.21365$ (GAS data) and $0.213\,47 \leqslant \beta_\infty \leqslant 0.213\,52$ (GABS data). This shows that $\mu = 4.6827 \pm 0.0022$ (GAS data) and $\mu = 4.683\,95 \pm 0.000\,55$. Rounding up these estimates show that

$$\mu = \begin{cases} 4.683 \pm 0.003 & \text{GAS;} \\ 4.6840 \pm 0.0006 & \text{GABS.} \end{cases} \tag{52}$$

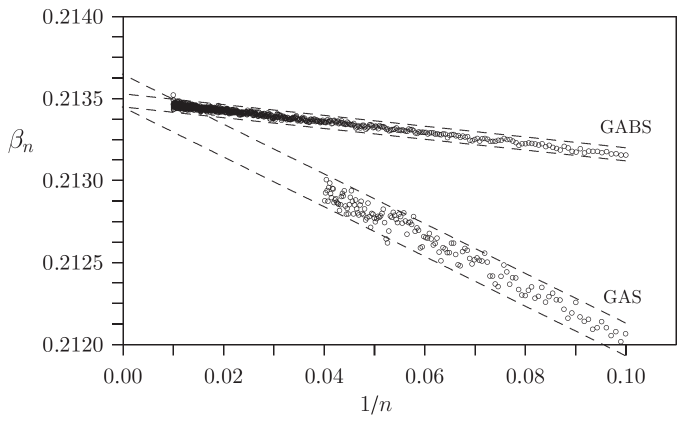

Figure 22. Extrapolating $\beta_n = [\langle a_-\rangle_n/\langle a_+\rangle_n]$ to $n \to \infty$ for generalized atmospheres (GAS) and endpoint atmospheres (GABS) in two-dimensional simulations of GAS and GABS. Since β_n should approach $1/\mu$ as $n \to \infty$, the Y-intercepts are estimates of $1/\mu$.

These results are comparable in accuracy to our best estimate in equation (50) above.

Overall the simulations using GABS were fast and provided reliable data for estimating critical exponents and growth constants. The implementation of GAS with generalized atmospheres is slower because each elementary move takes on average $O(n)$ CPU-time if the walk has length n, for $n \in [0, n_{\max}]$. This slowing down with increasing n of the algorithm can be overcome with better coding techniques, as was done for the pivot algorithm in [4].

Ideally, there should be the possibility of using more sophisticated series analysis methods with our data to extract better values for growth constants and entropic exponents. We are investigating this, but even longer simulations to obtained smoother data might be necessary. The extensions of GAS to other lattice models (such as polygons, trees, animals and surfaces) only require appropriate atmospheres in each case. We are currently investigating the sampling of knotted cubic lattice polygons with GAS.

Acknowledgments

EJJvR and AR acknowledge financial support in the form of Discovery Grants from NSERC (Canada). We are also grateful for valuable commentary on the paper by E Orlandini.

References

[1] 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

[2] Berg B and Foerster D 1981 Random paths and random surfaces on a digital computer *Phys. Lett.* B **106** 323–6

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

[4] Clisby N 2008 *Accurate Critical Exponents for Self-Avoiding Walks via a Fast Implementation of the Pivot Algorithm* in preparation

[5] Clisby N, Liang R and Slade G 2007 Self-avoiding walk enumeration via the lace expansion *J. Phys. A: Math. Theor.* **40** 10973–11017

[6] de Gennes P G 1979 *Scaling Concepts in Polymer Physics* (New York: Cornell University Press)

[7] Duplantier B 1986 Polymer network of fixed topology: renormalization, exact critical exponent γ in two dimensions *Phys. Rev. Lett.* **57** 941–4

[8] Flory P J 1949 The configuration of a real polymer chain *J. Chem. Phys.* **17** 303–10

[9] Flory P J 1955 Statistical thermodynamics of semi-flexible chain molecules *Proc. R. Soc. Lond.* A **234** 60–73

[10] Flory P J 1969 *Statistical Mechanics of Chain Molecules* (New York: Wiley-Interscience)

[11] Grassberger P 1997 Pruned-enriched Rosenbluth method: simulation of θ-polymers of chain length up to 1 000 000 *Phys. Rev.* E **56** 3682–93

[12] Guttmann A J and Conway A R 2001 Square lattice self-avoiding walks and polygons *Ann. Comb.* **5** 319–45

[13] Hammersley J M 1960 Limiting properties of numbers of self-avoiding walks *Phys. Rev.* **118** 656

[14] Janse van Rensburg E J and Rechnitzer R 2008 Atmospheres of polygons and knotted polygons *J. Phys. A: Math. Theor.* **41** 105002–25

[15] Jensen I 2004 Enumeration of self-avoiding walks on the square lattice *J. Phys. A: Math. Gen.* **37** 5503–24

[16] Le Guillou J C and Zinn-Justin J 1980 Critical exponents from field theory *Phys. Rev.* B **21** 3976–98

[17] Le Guillou J C and Zinn-Justin J 1985 Accurate critical exponents from the ϵ-expansion *J. Physique Lett.* **46** L137–L141

[18] Le Guillou J C and Zinn-Justin J 1985 Accurate critical exponents from field theory *J. Physique* **50** 1365–70

[19] 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

[20] 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

[21] Nienhuis B 1982 Exact critical point and critical exponents of $O(n)$ models in two dimensions *Phys. Rev. Lett.* **49** 1062–5

[22] Prellberg T and Krawczyk J 2004 Flat histogram version of the pruned and enriched rosenbluth method *Phys. Rev. Lett.* **92** 120602–5

[23] 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–L612

[24] Rechnitzer A and Janse van Rensburg E J 2008 Generalized atmospheric rosenbluth methods (GARM) *J. Phys. A: Math. Theor.* **41** 442002–10

[25] Rosenbluth M N and Rosenbluth A W 1955 Monte Carlo calculation of the average extension of molecular chains *J. Chem. Phys.* **23** 356–9

[26] 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