INTERNATIONAL COMPUTER SCIENCE INSTITUTE

1947 Center St. • Suite 600 • Berkeley, California 94704-1198 • (510) 643-9153 • FAX (510) 643-7684

Counting in Lattices: Combinatorial Problems from Statistical Mechanics

Dana Randall

TR-94-055

October 1994

Abstract

In this thesis we consider two classical combinatorial problems arising in statistical mechanics: counting matchings and self-avoiding walks in lattice graphs. The first problem arises in the study of the thermodynamical properties of monomers and dimers (diatomic molecules) in crystals. Fisher, Kasteleyn and Temperley discovered an elegant technique to exactly count the number of perfect matchings in two dimensional lattices, but it is not applicable for matchings of arbitrary size, or in higher dimensional lattices. We present the first efficient approximation algorithm for computing the number of matchings of any size in any periodic lattice in arbitrary dimension. The algorithm is based on Monte Carlo simulation of a suitable Markov chain and has rigorously derived performance guarantees that do not rely on any assumptions. In addition, we show that these results generalize to counting matchings in any graph which is the Cayley graph of a finite group.

The second problem is counting self-avoiding walks in lattices. This problem arises in the study of the thermodynamics of long polymer chains in dilute solution. While there are a number of Monte Carlo algorithms used to count self-avoiding walks in practice, these are heuristic and their correctness relies on unproven conjectures. In contrast, we present an efficient algorithm which relies on a single, widely-believed conjecture that is simpler than preceding assumptions and, more importantly, is one which the algorithm itself can test. Thus our algorithm is reliable, in the sense that it either outputs answers that are guaranteed, with high probability, to be correct, or finds a counterexample to the conjecture. In either case we know we can trust our results and the algorithm is guaranteed to run in polynomial time. This is the first algorithm for counting self-avoiding walks in which the error bounds are rigorously controlled.

This work was supported in part by an AT&T gradutate fellowship, a University of California dissertation year fellowship and Esprit working group "RAND". Part of this work was done while visiting ICSI and the University of Edinburgh.

Committee chair: Alistair Sinclair

Acknowledgments

I am extremely fortunate to have worked with Alistair Sinclair. His enthusiastic, inquisitive approach to research has left a deep impression and has certainly shaped my thinking about problems. Throughout every stage of this research he was a constant source of insight and encouragement.

Mike Luby has had a similar impact on my graduate school experience, often acting as my second advisor. Working with Luby has greatly influenced my appreciation for research. He has offered a lot of direction and advice, and his friendship has been one of my most important in Berkeley.

The faculty has made Berkeley a truly wonderful place to be a student. I have benefitted tremendously from classes and conversations with Dick Karp and Umesh Vazirani. Discussions with Manuel Blum fostered my curiosity and excitement during my initial stages of thinking about research. I am also grateful to Dick Karp and David Aldous for being on my thesis committee.

Claire Kenyon was wonderful to work with. I have learned much from her precise, delightful approach to research. I am very grateful to Rob Pike for implementing our self-avoiding walk algorithm, as well as for his friendship. Lars Rasmussen also deserves special thanks for implementing an earlier version of the algorithm.

Perhaps the greatest aspect of my graduate life at Berkeley was the exceptional student body. I have always turned to Diane Hernek with technical questions and have appreciated her generosity, her friendship and her humor. Will Evans was always available to bounce ideas off of and greatly enhanced my time here. And whenever the going got tough, Z Sweedyk helped me put things in perspective. Many of my most enjoyable times were spent with Eric Enderton, Amie Wilkinson, Sara Robinson and Ashu Rege. My graduate school experience would not have been complete without the friendships of Nina Amenta, Sandy Irani, Ronitt Rubinfeld, David Zuckerman, David Wolfe, Dan Jurafsky, Seth Teller, Leonard Schulman, Michael Mitzenmacher and the Stickhandlers.

Finally, for their love and support, I thank my family: Gladys, Richard, Barbara and Lisa Randall, and Lori Wood.

Contents

List of Figures

Chapter 1

Introduction

Statistical mechanics provides a rich source of fundamental combinatorial questions with natural applications. Physicists use sophisticated algorithms for these problems in order to understand various physical systems, but often these algorithms are non-rigorous or inefficient. With the recent developments in computer science for designing efficient algorithms with rigorous performance guarantees, there is an increasing demand for the exchange of ideas to tackle these combinatorial problems.

The standard scenario is as follows. There is a set of combinatorial structures corresponding to the allowable configurations of a physical system. The primary challenge is to *count* the number of configurations. The second is to *sample* a configuration from the set at random. Solutions to these problems would provide valuable insight into the related physical systems.

This thesis focuses on two types of combinatorial structures: matchings and self-avoiding walks in lattices. These are classical problems which arise in the context of monomer-dimer systems and long polymers chains. We present the first provably efficient approximation algorithms for solving the counting and sampling problems in each case. The algorithms are based on Monte Carlo simulations similar to those widely used in statistical mechanics, and the analysis uses insights from computer science for deriving rigorous statistical bounds on the accuracy of such simulations. A feature which distinguishes these results from other recent work in the area is that we use the special structure of the lattice in a critical way. Lattices represent precisely the set of graphs for which there is the greatest physical significance.

The following sections are intended both to motivate the work in this thesis and to provide some of the essential background. Section 1.1 gives a brief overview of the statistical mechanics applications; section 1.2 places these physical problems in the framework of computational complexity theory; finally section 1.3 formalizes the definitions of approximation algorithms and gives a basic outline of the algorithmic machinery used.

1.1 Physical systems

Statistical mechanics endeavors to relate the observable, macroscopic properties of a physical system, such as density and temperature, to the microscopic interactions among particles of the system. The goal is to understand how simple, local interactions between small numbers of particles determine the macroscopic behavior and to predict how various parameters of the system contribute to these observable effects.

The complicated, dynamic interplay among large numbers of particles in a system makes a precise characterization of all microstates impossible, so state information is captured as a probability distribution over all feasible *configurations* of particles; a configuration captures the essential information of a microstate and is assigned a probability according to the likelihood of the associated microstates. The *partition function* Z, which is a weighted sum over the set of possible configurations, is the key to relating the two levels of description. Most of the thermodynamic properties describing the macrostate of a system can be derived from knowledge of Z. The goal in this thesis will be to develop tools which will help us compute close approximations to the partition function for some classical physical systems.

1.1.1 The partition function

A configuration of a system is a description of a possible microstate. For example, consider a system with N particles in volume V held at a constant temperature T through interaction with some external heat source. A configuration includes the positions of the N particles as well as any other relevant information such as their magnetic moments. The system is modeled discretely so that the positions of atoms coincide with vertices of a finite lattice. This can be thought of as a finite $n \times n \times ... \times n$ subset of the cartesian lattice $\mathcal{Z}^d$ or a finite subgraph of some other regular lattice.

Associated with each configuration is an *energy*. The probability distribution over configurations is defined by a simple function of this energy. The form of this function depends on which configurations are consistent with the fixed parameters of the system, in the above example (N, V, T). We describe this distribution for three typical families of systems.

In the simplest case of an isolated system, the internal energy is held constant. All configurations which have this energy E are considered equally likely and all others are considered impossible. This probability distribution over configurations with fixed values fo the parameters (N, T, V, E) is known as the *microcanonical* distribution.

In the more interesting cases with which we will concern ourselves in this thesis, the system is not isolated but interacts with some external source. In the first case the parameters (N, V, T) are held constant, but the energy of the system is allowed to vary. The likelihood of a particular configuration is given by the *canonical* (or *Boltzmann*, or *Gibbs*) distribution. Again all configurations with equal energy are equally likely, and now the probability of different configurations is exponentially distributed over energies. More precisely, let G denote the lattice graph and let $E(s)$ be the energy of a configuration s. The probability of s is

$$\pi(s) = \exp(-E(s)/kT)/Z,$$

where k is *Boltzmann's constant* and Z is the normalizing constant,

$$Z \equiv Z(G; N, V, T) = \sum_s \exp(-E(s)/kT). \tag{1.1}$$

The weighted sum over all configurations given in equation (1.1) is known as the *partition function* for the canonical distribution.

A third common distribution arises in physical systems which interact with a permeable membrane allowing the number of particles N to vary. The likelihood of a configuration with a fixed number of particles is controlled by a parameter x known as the *activity* or *fugacity* of the system. For fixed (x, V, T), the *grand canonical* distribution is a function in the parameter x. The probability of a configuration s with $|s|$ particles is given by

$$\pi(s) = \frac{\exp(-E(s)/kT) \cdot x^{|s|}}{Z},$$

where again the normalizing constant Z is the partition function for this distribution. If Z_N is the partition function for the canonical distribution where the number of particles is fixed to be N and the number of vertices in the lattice graph G is $\hat{N}$, then we can write the grand canonical partition function as

$$Z \equiv Z(G; x, T) = \sum_s \exp(-E(s)/kT) \cdot x^{|s|} = \sum_{N=0}^{\hat{N}} Z_N \ x^N. \tag{1.2}$$

Computing the partition function of a system is a primary objective of statistical mechanics. The significance of Z stems from the fact that many of the thermodynamic properties of a physical system are related to $\log Z$ or one of its derivatives; we will illustrate this relationship in the next subsection. For computational purposes, it is often useful to view the partition function as a generating function. This is possible when the energy levels are discrete. Specifically, let $\mathcal{E}$ be the set of all possible values for the energy of a configuration, and let a_E be the number of configurations with a particular energy $E \in \mathcal{E}$. Let $y = \exp(-1/kT)$. When the number of particles N is fixed, the partition function for the canonical distribution defined in equation (1.1) is just the generating function

$$Z = \sum_{E \in \mathcal{E}} a_E \ y^E.$$

When the number of particles varies, then we let $a_{N,E}$ be the number of configurations with N particles and energy E. The partition function for the grand canonical distribution from equation (1.2) is

$$Z = \sum_{E \in \mathcal{E}} a_{N,E} \ y^E \ x^N.$$

Computing the partition function in these cases can be reduced to computing the coefficients a_E or $a_{N,E}$ of the relevant generating function.

We now give three classical examples of physical systems to demonstrate these concepts. The problems are defined for any finite lattice and easily generalize to any finite graph. In each of the following cases it is convenient to think of the special case of an $n \times n$ rectangular lattice (or chessboard).

Example 1: The Ising Model

The Ising model was introduced in the 1920's to study ferromagnetism and is one of the most famous models from statistical mechanics (see, e.g., [8] for a history and review). The system is modeled by a finite lattice where the vertices represent atoms of a ferromagnetic material. Each configuration σ consists of an assignment of a +1 or -1 spin σ_i to each of the vertices i, representing the magnetic moment of each atom. In the Ising model nearest neighbors tend to align with each other according to a parameter $J > 0$, and all the atoms tend to align with an external magnetic field H. The energy of a configuration σ is given by

$$E(\sigma) = -\sum_{(i,j)} J\sigma_i\sigma_j - \sum_i H\sigma_i,$$

where the first summation is taken over nearest neighbor pairs. From equation (1.1) the partition function for the canonical distribution is then

$$Z = \sum_\sigma \exp(-E(\sigma)/kT).$$

```
− − − − + + + + + + + +        + − + + + − − + + − + −
− − − − + + + + + + + +        + + − + − + + − + − + +
− − − + + + + + + + + +        − − − + + − − + − + − +
− − + + + + − − − + + +        + + + − + − − + − + + −
− − + + + − − − − − + +        − + + + − + + + − − + −
+ + + + − − − − − − + +        + − + − + − + − + + − +
+ + + + + + − − − − + +        − + + − − − + + − + − +
+ + + + + + + − − + + +        + − + − + + − + − + + −
+ + + + + + + + + + + +        + + − − + + − + + − + +
+ + + + + + + + + + + +        − + + − + − + + − + − +
+ + + + + − − − − + + +        + + + − − + + − + + + −
+ + + + − − − − − + + +        + − + − + + − + + − + +
```

Low temperature (with $H > 0$)   High temperature

Figure 1.1: Examples of typical Ising configurations at low and high temperatures.

At low temperatures the parameters J and H will have greater influence and we are more likely to see large clusters of like spins. At higher temperatures there will tend to be less organization. Figure 1.1 shows typical configurations for each of these cases.

The Ising model can also be used to represent *lattice gases*, where a +1 indicates that a lattice site is occupied by a molecule and a -1 indicates it is vacant. The high and low temperature diagrams in figure 1.1 reflect the fact that at higher temperatures the gases move freely and particles do not influence their neighbors much, while at lower temperatures particles attract each other and will tend to cluster as in the solid state. The partition function provides information about the phase transition between gaseous and solid states.

Example 2: Monomer-Dimer Systems

Another fundamental challenge in chemical physics is the *monomer-dimer* problem in which the sites of a regular lattice are covered by a non-overlapping arrangement of *dimers* and *monomers*. A dimer is a diatomic molecule which covers two adjacent vertices in the lattice, and a monomer covers each vertex not covered by a dimer (see, e.g., [30] for a

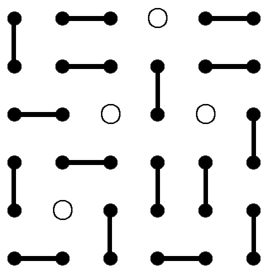


Figure 1.2: A monomer-dimer arrangement.

history of the problem). In graph theoretic terms, the monomer-dimer arrangement is just a *matching*; a matching of size i is a set of i lattice edges such that no two edges share a vertex. Figure 1.2 shows a typical monomer-dimer arrangement with 16 dimers and 4 monomers. The two-dimensional monomer-dimer problem serves as a model for the adsorption of diatomic molecules onto a crystal surface, where monomers correspond to empty sites [56]. The three-dimensional problem occurs classically in the theory of mixtures of molecules of different sizes [20] and the cell-cluster theory of the liquid state [9].

Consider a monomer-dimer arrangement consisting of s dimers and $\hat{N} - 2s$ monomers on a lattice with $\hat{N}$ vertices. The energy is sJ, where J is the energy derived from covering any particular edge with a dimer. Let $\mathcal{M}_s$ be the set of monomer-dimer arrangements with s dimers, and let $|\mathcal{M}_s|$ be the cardinality of this set. The partition function for the canonical distribution with a fixed number of dimers s is

$$Z_s \equiv Z_s(G; N, T) = \sum_{M \in \mathcal{M}_s} \exp(-sJ/kT) = |\mathcal{M}_s| \exp(-sJ/kT).$$

The grand canonical distribution describes more interesting systems where the number of dimers varies. If x is the activity of a dimer, then following equation (1.2) the grand canonical distribution is

$$Z \equiv Z(G; x, T) = \sum_{s=0}^{\lfloor \hat{N}/2 \rfloor} \exp(-sJ/kT) x^s = \sum_{s=0}^{\lfloor \hat{N}/2 \rfloor} |\mathcal{M}_s| \exp(-sJ/kT) x^s.$$

Letting $\mu = \exp(-J/kT) \cdot x$ gives gives the generating function

$$Z \equiv Z(G; \mu) = \sum_{s=0}^{\lfloor \hat{N}/2 \rfloor} |\mathcal{M}_s| \, \mu^s. \tag{1.3}$$

Evaluating the coefficients of this generating function is exactly the problem of counting the numbers of monomer-dimer arrangements with given numbers of dimers.

Example 3: Lattice Animals

A *lattice animal* is a set U of vertices on the lattice such that the subgraph induced by U is

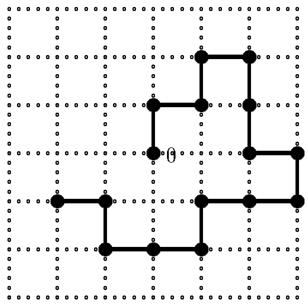


Figure 1.3: A self-avoiding walk.

connected (see [67]). An important subclass of animals are *self-avoiding walks*, or animals where each vertex has degree at most two. Equivalently, a self-avoiding walk of length i starts at a fixed origin and follows the lattice edges for i steps with the constraint that it never visits the same lattice site twice (see figure 1.3). The partition function here is a generating function where the coefficients c_i are the number of walks length i. Letting $\hat{N}$ be the number of vertices in the lattice and taking x as the monomer fugacity, the grand canonical partition function from equation (1.2) is

$$Z = \sum_{w} x^{|w|} = \sum_{i=1}^{\hat{N}} c_i x^i.$$

The self-avoiding walk models a dilute solution of long polymer chains in a good solvent. This model arises since the polymer chain can be thought of as tracing out a random walk in space, except that the physical restriction that no two molecules can occupy the same position forces the self-avoidance condition. The dilute solution enables one to study a single polymer without worrying about further restrictions caused by interactions with other polymer chains (see [48] for a survey). The more general class of lattice animals arises similarly in the study of branched polymers in dilute solution.

As we have already stated, for each of these physical systems, a primary objective is to calculate the partition function. By interpreting this function as a generating function, we can instead study the underlying combinatorial problems which are intrinsically tied to these systems. The typical problem then becomes one of computing the coefficients of the partition function; this will generally entail counting the number of configurations in a set (e.g. the number of self-avoiding walks of length i). We will also be interested in the problem of randomly sampling these configurations s according to their likelihood, $\pi(s)$, in the appropriate canonical distribution. Sampling allows us to examine "typical" configurations of the system. Moreover, it has been recognized that for large classes of combinatorial problems the counting and sampling problems are closely related [36]. For each of the problems we consider we will be addressing both of these questions.

1.1.2 Free energy

As indicated earlier, the partition function Z is the key to relating the microscopic and macroscopic levels of description of a physical system. While the partition function is defined as a sum over microstates, it captures the essential information determining the macroscopic behavior of a system. In particular, the *free energy* $F = -kT \log Z$ is a thermodynamic primitive from which most other thermodynamic properties can be derived.

To appreciate the role of the free energy, first consider its relation to other well-studied macroscopic functions, the *entropy* and the *average energy* of a system. The entropy, which is defined as $S = \sum_s \pi(s) \log \pi(s)$, represents a measure of the disorder (or "unavailable energy") of a closed system. For the canonical distribution (equation (1.1)),

$$\begin{aligned} kTS &= \sum_s E(s) \frac{\exp(-E(s)/kT)}{Z} - kT \log Z \\ &= \overline{E} - F, \end{aligned}$$

where $\overline{E}$ is the expected (or average) energy of the system.

More significantly, we can derive information about many thermodynamic properties of a physical system by studying how the free energy changes due to small perturbations in the fixed parameters of the system. An important example is the partial derivative with respect to temperature. Discontinuities in this derivative identify *phase transitions.* In the ferromagnetic interpretation of the Ising model, such a phase transition corresponds to the point of spontaneous magnetization. Other examples of phase transitions are melting and boiling (see [8]).

In our discussions so far we have restricted our attention to idealized finite systems where the particles coincide with lattice sites. In fact, these discrete models are really computational tools for studying the continuous analogues of the partition function and free energy on infinite lattices. The actual thermodynamic properties of a system are realized by considering successively larger lattices and studying the behavior as the lattice size tends to infinity. This is known as the *thermodynamic limit.* In this limit, observations regarding the behavior of the free energy determine properties of the actual system under study. In most cases there is no known way to directly compute the asymptotic behavior of thermodynamic functions, and defining discrete analogues to these functions appears to be one of the more promising approaches. Calculating the partition function for various finite lattices therefore provides valuable insight into the corresponding real physical systems.

1.1.3 The model

Typically we think of the lattice as a finite $n \times n \times ... \times n$ subgraph of the cartesian lattice $\mathcal{Z}^d$, although other families of lattices are also studied. In some cases it will be more appropriate to talk about finite configurations on either an infinite lattice or a periodic lattice where opposite sides are identified. The choice of the lattice will be clear from the application.

As explained above, it is generally accepted that modeling various systems by families of successively larger lattices will tell us about the limiting continuous behavior. However, it is not immediately clear why our choice of combinatorial structures is a realistic model of the physical constraints of the particles in the systems. In particular, self-avoiding walks on the cartesian lattice appear to have a quite different set of constraints from polymers, whose bond angles are rarely rectilinear. In fact the bond angles are typically tetrahedral and do not align with any regular lattice [61].

The justification for using self-avoiding walks to model polymers, and for the other combinatorial idealizations employed in the field, comes from the empirical existence of *universality* classes. Several of the parameters studied in the thermodynamic limit appear to be dimension dependent, but lattice independent. This means that to study these universal quantities for any particular member of the class, including physically realistic models for polymers, it is sufficient to determine them for any other member. Therefore, the mathematically simpler model of self-avoiding walks on lattices lets us deduce these thermodynamic properties for the more realistic polymer models.

1.2 Computational complexity of counting problems

Viewing the partition function as a generating function has identified several combinatorial problems as essential ingredients in understanding a physical system. Recall that when the energy is discrete the partition function has the form $\sum_i a_i y^i$, where the coefficient a_i is the cardinality of some set of configurations $\mathcal{S}_i$. For example, in the partition function of a monomer-dimer system, $\mathcal{S}_i$ represents monomer-dimer arrangements with i dimers and a_i is the number of such arrangements (see equation (1.3)). The primary combinatorial questions fall into two categories:

(i) Counting: e.g., calculate a_i, the number of elements in $\mathcal{S}_i$.

(ii) Sampling: e.g., pick an element from $\mathcal{S}_i$ uniformly to determine the expected value of some function over the elements of $\mathcal{S}_i$.

As explained earlier, the *counting* problem is used to compute the coefficients of the partition function. The *sampling* problem is used to gain information about a "typical" configuration. For example, in a monomer-dimer arrangement with $\hat{N}/2 - 1$ dimers, one might be interested in studying the expected distance between the two monomers.

Alternatively, it is sometimes more appropriate to try to solve the weighted analogues of these problems. Here we have a set $\mathcal{S}$ where each element $s \in \mathcal{S}$ has an associated weight $w(s) = \exp(-E(s)/kT)$ and we are interested in the following questions:

(i) Evaluate the weighted sum $\sum_{s \in \mathcal{S}} w(s)$.

(ii) Generate an element of $\mathcal{S}$ at random according to its weight.

Solving the weighted counting problem would allow us to calculate the partition function directly, although we would not know all of the individual coefficients. Both the unweighted and weighted versions of the counting and generation problems will fall into a similar framework.

Ideally we would like analytic solutions to the counting (or weighted counting) problems. In 1944, Onsager discovered such a closed-form expression for the partition function of the two-dimensional Ising model with zero external field [53]. It has not been possible to find such a precise solution for most other physical systems, or even for the Ising model in higher dimensions or in the presence of an external field.

Consequently most of the research in recent years has been concentrated on designing efficient *algorithms* to solve the counting and generation problems. Algorithms based on Monte Carlo simulations, in particular, have provided great insight into various physical systems. However, while they are useful tools, many of the algorithms actually used in statistical mechanics are nonrigorous applications of the methods and are not reliable. Recent progress in the design of provably accurate and efficient approximation algorithms enables us to develop new Monte Carlo algorithms with guaranteed error bounds.

1.2.1 Efficient algorithms

For an algorithm to be practical, it must be *efficient* in the sense that its running time does not grow too fast with the input size. The generally accepted formalization of the notion of efficiency is a *polynomial-time algorithm.* Such an algorithm takes as input a natural description of the problem instance and must compute an output in time which is bounded by some fixed polynomial in the size of the input description (see, e.g., [31]). For problems in statistical mechanics involving lattices, the input size is taken to be $\hat{N}$, the number of vertices in the lattice; a polynomial-time algorithm (e.g., for the counting or generation problems) must solve the problem in $O(\hat{N}^k)$ time, for some fixed k. This is a natural measure of the input size since the number of solutions is exponential in $\hat{N}$ and it requires $O(\hat{N})$ steps just to write down a typical configuration.

Note that, for any of the physical problems presented in section 1.1, it is trivial to design an exponential-time algorithm which computes the output in time $\exp(O(\hat{N}))$. Since the number of configurations on a lattice of size $\hat{N}$ is at most exponential in $\hat{N}$, we can do this by exhaustively enumerating each configuration. Of course such an algorithm is not polynomial-time and is impractical unless the size of the lattice is very small. We should also note that polynomial-time algorithms are not necessarily efficient in practice; for polynomials with large degree it might be infeasible to run the algorithm except for small inputs. However, a polynomial-time algorithm is certainly a dramatic improvement over a trivial exponential-time one, and typically such algorithms are later improved so as to be genuinely practical.

A breakthrough in the design of efficient algorithms in statistical mechanics was achieved in 1961 when Fisher, Kasteleyn and Temperley independently discovered a polynomial-time algorithm for a special case of the monomer-dimer problem known as the *dimer problem* [15,40, 63] (see section 2.1). The dimer problem asks for the number of *perfect matchings*, or coverings of the lattice by $\hat{N}/2$ dimers (and no monomers). (More generally, a perfect matching in a graph is a matching in which every vertex of the graph is incident to some edge in the matching.) The Fisher, Kasteleyn, Temperley technique generalizes to counting perfect matchings in any planar graph and can also be used to construct a polynomial-time algorithm for the partition function of the two-dimensional Ising model with zero external field.

Despite major efforts to generalize these techniques, it does not appear that they can be used to solve the monomer-dimer problem in the presence of monomers or the Ising model in non-zero field, even in two dimensions. Similarly, the techniques rely critically on planarity and do not appear to generalize to higher dimensional lattices. The first formalization of this limitation came when Hammersley et al. [25] proved that there cannot be a straightforward way of extending the above approach to even three-dimensional lattices.

1.2.2 Hardness

Computational complexity theory offers further evidence for the apparent hardness of these problems. In 1979, Valiant defined the class #P to classify counting problems [64]. A counting problem asking for the number of elements in a set $\mathcal{S}_n$ representing configurations on a graph G belongs to #P if there is a non-deterministic polynomial-time Turing machine with input n and G such that the number of accepting computations is exactly $|\mathcal{S}_n|$, the number of elements in $\mathcal{S}_n$. Note that this is exactly the framework for the class NP; while NP is the set of problems for which the non-deterministic machine decides whether an accepting computation *exists*, #P is the class of problems for which the non-deterministic machine counts *how many* accepting configurations exist (see [18] for a survey).

Each of the computational problems from the previous section can be expressed as a general graph theoretic problem. For instance, the problem #MATCHING(G, i) takes as input a graph G and an integer i and asks how many matchings of size i exist. This problem is in #P since we can construct a non-deterministic Turing machine which "guesses" a set of edges of G and then checks, in time polynomial in $\hat{N}$ (the number of vertices of G) whether this is actually a valid configuration. Similarly, #SAW(G, i, x_0) takes as input a graph G, and integer i and a designated vertex x_0 and asks for the number c_i of self-avoiding walks of length i starting at the vertex x_0. Finally, #ISING(G, E) takes as input a graph G and asks how many configurations have a particular energy E. These problems are also in #P since we can design a non-deterministic algorithm which guesses a candidate configuration and then decides, in polynomial time, whether this configuration should be included in the count.

The class #P has a subclass of *complete* problems which characterize, in a precise sense, the hardest problems in the class. To formalize this we need the following definition. For any two counting problems A and B, A is *polynomial-time Turing reducible* to B if there exists a polynomial-time algorithm for A which uses B as a subroutine. Thus, if A is polynomial-time reducible to B, then the existence of a polynomial-time algorithm for B implies that one also exists for A. A problem B is *#P-complete* if every problem in #P is polynomial-time reducible to B. Thus, showing that *any* #P-complete problem can be solved in polynomial time would imply that there is a polynomial time algorithm for *every* problem in $\#P$. #P-completeness is regarded as strong evidence of intractability: exhibiting a polynomial time algorithm for any #P-complete problem would provide efficient solutions to many hard problems such as counting the number of satisfying assignments of a boolean formula. Since it is expected that no polynomial-time algorithm can even decide whether a satisfying assignment exists, it is even more unlikely that an efficient algorithm exists to *count* the number of satisfying assignments.

Valiant showed that #MATCHING$(G, \hat{N}/2)$, the problem of counting perfect matchings in a graph with $\hat{N}$ vertices, is #P-complete [64]. This helps explain why Fisher, Kasteleyn and Temperley's algorithm fails to generalize to all non-planar graphs. Furthermore, Jerrum showed that counting the total number of matchings (of all sizes) in a graph is #P-complete, even when the graph is planar [32]. This gives evidence for why the Fisher, Kasteleyn, Temperley algorithm does not easily generalize to counting matchings of arbitrary size. Similarly, consider the problem $\#\mathrm{SAW}(G, \hat{N}-1, x_0)$ which asks us to count self-avoiding walks of length $\hat{N}-1$ on a graph G with $\hat{N}$ vertices starting from x_0. This is just the problem of counting the number of Hamiltonian paths starting at x_0, another problem known to be #P-complete.

Of course we cannot claim that these #P-complete problems, which are hard when given an arbitrary graph as input, remain hard when the input graph is restricted to be a lattice. Nonetheless, these hardness results do suggest that any polynomial-time algorithm to solve these problems must use properties of the lattice in a non-trivial manner. No polynomial-time algorithm which does this is known. In fact, exploiting lattice properties to design efficient algorithms has proven to be notoriously difficult.

While initially discouraging, the #P-completeness of many important combinatorial problems has shifted focus towards designing efficient *approximation algorithms*. An approximation algorithm uses randomization to produce a close estimate to the true answer with high probability. Recently there has been much progress in the design and analysis of efficient randomized algorithms for approximately counting. Approximation algorithms are generally sufficient for combinatorial problems arising from statistical mechanics since we are interested in studying the limiting behavior of quantities associated with the system, and these algorithms allow us to approximate the true values to arbitrary precision.

1.3 Approximation algorithms

Approximation algorithms based on computer simulations of a random process have assumed an important role for a wide range of combinatorial problems. The idea is as follows. Let $\mathcal{S}$ be a large but finite set of combinatorial structures. Much information about $\mathcal{S}$ can be gained by sampling elements of $\mathcal{S}$ according to an appropriate probability distribution π. For example, suppose that π is chosen to be the uniform distribution. At least intuitively, sampling elements uniformly is useful for studying properties of a typical element of $\mathcal{S}$. In fact, sampling uniformly turns out to be a useful tool for approximate counting as well.

More formally, we will be interested in designing algorithms which meet the following specifications. These are the standard definitions of approximation algorithms for counting and sampling (see, for example, [39, 36, 59]).

Let $\mathcal{S}_n$ be the set of configurations of size n on a lattice G of size $\hat{N}$ (for example, $\mathcal{S}_n$ might be the set of self avoiding walks of length n on an $\hat{N}^{1/2} \times \hat{N}^{1/2}$lattice). Let a_n be the number of elements in $\mathcal{S}_n$.

Definition 1.3.1 *A* randomized approximation scheme *for a_n on a lattice G of size $\hat{N}$ is a probabilistic algorithm which, on input n, G and $\epsilon, \delta \in (0,1)$, outputs a number $f(G,n)$ such that* $\Pr\{a_n(1+\epsilon)^{-1} \leq f(G,n) \leq a_n(1+\epsilon)\} \geq 1-\delta$. *The approximation scheme is* fully-polynomial, *or an* fpras, *if it is guaranteed to run in time polynomial in $\hat{N}, \epsilon^{-1}$ and* $\log \delta^{-1}$.

Definition 1.3.2 *An* almost uniform generator *for $\mathcal{S}_n$ on a lattice G of size $\hat{N}$ is a probabilistic algorithm which, on input n, G and $\epsilon \in (0,1)$, outputs an element σ of $\mathcal{S}_n$ with probability at least $1/q(\hat{N})$ for a fixed polynomial q, such that the conditional probability distribution over elements of $\mathcal{S}_n$ has variation distance* at most ϵ from the uniform distribution. The generator is* fully-polynomial, *or an* fpaug, *if it runs in time polynomial in $\hat{N}$ and* $\log \epsilon^{-1}$.

The parameter ϵ determines the accuracy required of the estimate, while δ controls the confidence level. A fully-polynomial randomized approximation scheme provides an efficient means of numerically computing f, in the sense that its running time grows only *slowly* (i.e., polynomially) with the lattice size $\hat{N}$, the accuracy parameter ϵ, and the confidence parameter δ. Similarly, a fully-polynomial almost uniform generator gives an efficient means for solving the sampling problem.

In the cases we are interested in, $\mathcal{S}_n$ represents some exponentially large set of combinatorial objects such as self-avoiding walks of length n, so naive methods based on exhaustive enumeration are infeasible. Instead we will focus on designing an efficient solution to the corresponding sampling problem and then show how this will provide a tool for solving the counting problem.

Suppose we want to sample from $\mathcal{S}_n$ according to some distribution π. The standard approach is to simulate a Markov chain whose state space Γ includes $\mathcal{S}_n$. At each point in time t we visit a combinatorial structure $s \in \Gamma$. In the next time step, we move to a (possibly) new structure by performing some random local perturbation to the structure s.

For example, let $\mathcal{S}_n$ is the set of self-avoiding walks of length n and suppose we want to sample a walk in $\mathcal{S}_n$ uniformly. We can let Γ be the set of self-avoiding walks of length at most n, and choose a distribution π' on Γ such that the conditional probability of π' on $\mathcal{S}_n$ is the uniform distribution. To simulate the Markov chain we start at the empty walk w_0. At any point in time, if we are currently at a self-avoiding walk w, then at the next step we move, with some appropriately chosen probability, to a walk $w' \in \Gamma$ which differs from w by at most one edge. After iterating this process for a sufficient number of steps $\hat{t}$, we examine the walk $w_{\hat{t}}$. If $w_{\hat{t}} \in \mathcal{S}_n$, then we output it; otherwise we start again. In order for this method to satisfy the requirements of a fully-polynomial almost uniform generator (see definition 1.3.1), the final distribution from which we are sampling must be reasonably well concentrated on $\mathcal{S}_n$ (so that one gets a valid sample quite often), and the Markov chain must converge rapidly to its stationary distribution (so that the number of simulation steps required is not too large).

Guaranteeing all of these conditions for a particular Markov chain can be difficult. Already Monte Carlo algorithms based on Markov chains appearing to satisfy these properties are used extensively in the physical sciences. These simulations have provided great insight into the asymptotic behavior of physical systems, but they are often nonrigorous. The most common difficulty arises from not knowing how long to simulate the Markov chain. A sample taken after simulating the Markov chain for too few steps might be chosen according to an unknown distribution which is quite far from the stationary distribution. Consequently,

*The variation distance measures the distance between two distributions ν_1, ν_2 over $\mathcal{S}_n$ and is defined as $\|\nu_1 - \nu_2\| = \frac{1}{2}\sum_{s \in S} |\nu_1(s) - \nu_2(s)| = \max_{A \subseteq S} |\nu_1(A) - \nu_2(A)|$.

statistics inferred from such samples might appear to have similar properties to those being tested and yet the results would in fact say nothing about the true distribution being studied. Therefore it is necessary to rigorously show that the Markov chain has the desired properties and is simulated a sufficient number of steps.

1.3.1 Markov chains

All of the Markov chains we will be using are variants of the Metropolis algorithm discovered in 1953 [49]. We will review some of the concepts of Markov chains in the context of this algorithm. Again, assume we have a set $\mathcal{S}$ from which we want to sample according to a distribution π. We define the Markov chain on a possibly larger state space Γ containing $\mathcal{S}$. First we choose a graph H underlying the Markov chain whose vertices are the states in Γ. Every element $i \in \Gamma$ has a small number $d \leq \hat{d}$ of neighbors in H. Furthermore, assume that for any edge (i,j) in H, we have an "acceptance probability" $A(i,j) \in (0,1]$.

The Markov chain is designed so that if we take a random walk along the edges of H according to the transition probabilities starting at any state, then we will eventually converge to a *stationary* (or *equilibrium*) distribution π' over the state space Γ. We choose the graph H and the transition probabilities A so that the stationary probability of being in $\mathcal{S} \subseteq \Gamma$ is not too small, and so that the conditional probability $\pi'(s)/\pi'(S)$ of being at any particular state $s \in \mathcal{S}$ is $\pi(s)$. If we know the stationary distribution in advance, then by simulating the Markov chain for sufficiently many steps, we can sample according to this distribution.

We always choose the graph H so that it is connected; i.e., it is possible to get from any state of the Markov chain to any other state; such a Markov chain is called *irreducible*. If, in addition, we know that there is a time t when it has positive probability of being at each state, then the chain is *aperiodic*. A Markov chain with both of these properties is *ergodic* and there is a unique stationary distribution π' to which the Markov chain converges.

The transition probabilities are defined as follows. Starting at an arbitrary point $i \in \mathcal{S}$ we perform the following steps:

(i) Choose a random neighbor j of i, each with probability $1/\hat{d}$.

(ii) Move to j with probability $A(i,j)$; otherwise stay at i.

Then the *transition probabilities* are represented by a matrix P where

$$P(i,j) = A(i,j)/\hat{d},$$

if $i \neq j$, and

$$P(i,i) = 1 - \sum_{j \neq i} A(i,j)/\hat{d}.$$

In the Metropolis algorithm we define $A(i,j) = \min(1, \pi'(j)/\pi'(i))$. It is then easy to verify that $P(i,j)$ satisfies the *detailed balance* equation

$$\pi'(i)P(i,j) = \pi'(j)P(j,i). \tag{1.4}$$

Informally, the detailed balance equation says that at stationarity the Markov chain is equally likely to move from i to j as the other way around. A Markov chain which satisfies this condition for all states i and j is called *reversible*, and π' is the unique stationary distribution to which the Markov chain converges, starting at *any* initial state. All the Markov chains we will be using are reversible.

This framework lets us design Markov chains which will eventually converge to a particular conditional distribution π on our state space $\mathcal{S}$. For this to be useful for efficiently sampling according to π, we need to bound the rate of convergence. This property is called the *mixing rate* of the Markov chain. Since the state space is typically exponentially large (in the description of the combinatorial problems we are using it for), we need a strong condition which says that after only a polynomial number of steps we are close to stationarity. This is called *rapid mixing* and can be characterized by the *expansion properties* or *conductance* of the graph underlying the Markov chain [59, 34]. We will explain this connection in greater detail in section 3.2.2.

1.4 Summary

The remainder of this thesis is divided into two chapters. The first chapter addresses the monomer-dimer problem. We show that there is a fully-polynomial randomized approximation scheme for counting the number of matchings of any cardinality on any periodic finite lattice (or, more generally, any Cayley graph.) A periodic lattice includes "wrap-around" edges which make the lattice into a torus. These results also extend to counting matchings on non-periodic planar lattices. This gives an approximation algorithm which solves the monomer-dimer problem in exactly the two cases where the Fisher, Kasteleyn, Temperley algorithm fails: namely, counting matchings of arbitrary size, and counting matchings on non-planar lattices. These results are based on an approximation algorithm due to Jerrum and Sinclair [34] and Broder [7]. In each of these cases we also present an almost uniform generator for the corresponding sampling problems. This chapter is based on work with Claire Kenyon and Alistair Sinclair [43].

In chapter 3 we address the problem of counting self-avoiding walks in lattices. We present a "testable algorithm" to solve this problem. We introduce the notion of a *testable algorithm* to describe an efficient algorithm whose correctness is based on a conjecture which the algorithm systematically verifies. The algorithm either discovers a counterexample to the conjecture or produces numerical answers which are correct with high probability. Therefore, any outputs produced by the algorithm are correct. We design such an algorithm for counting and generating self-avoiding walks in any finite rectangular lattice in arbitrary dimension. The algorithm relies on a single conjecture which is widely accepted in the physics community. Consequently, we expect that the algorithm will always output correct numerical outputs; on the other hand, should the algorithm find that the conjecture is incorrect, this would have interesting implications as well. These are the first algorithms for counting and generating self-avoiding walks where the error bounds are rigorously controlled. These results are based on work with Alistair Sinclair [54]. An optimized version of the algorithm was implemented by Rob Pike yielding numerical estimates for the number of self-avoiding walks in 2 and 3 dimensions; these estimates are presented in section 3.5.

Chapter 2

Matchings

2.1 Introduction

2.1.1 Historical background

The first problem we consider is the *monomer-dimer* problem, in which the sites of a regular lattice are covered by a non-overlapping arrangement of monomers (molecules occupying one site) and dimers (molecules occupying two sites that are neighbors in the lattice). For a given monomer-dimer arrangement with d dimers and $\hat{N} - 2d$ monomers on a lattice of size $\hat{N}$, the *dimer density* is defined as $2d/\hat{N}$. We are interested in counting the number of monomer-dimer arrangements with any fixed density. In this section we present some of the highlights in the history of this problem. For further information see, e.g., [30, 42, 66] and the references given there.

The monomer-dimer problem gained prominence in 1937 through the early paper of Fowler and Rushbrooke [17]. A breakthrough was achieved in 1961, when, independently, Fisher, Kasteleyn and Temperley provided an analytic solution for the case of *dimer coverings* (i.e., arrangements with dimer density 1) on a two-dimensional rectangular lattice [15, 40, 63]. The key idea is to express the number of dimer coverings as a Pfaffian, which in turn can be evaluated as the square root of an associated determinant. These calculations give precise asymptotics for $f(n)$, the number of dimer coverings of an $n \times n$ rectangular lattice (with n even); specifically,

$$\frac{1}{n^2} \ln f(n) \to \lambda \text{ as } n \to \infty, \text{ where } \lambda = \frac{1}{\pi} \sum_{r \geq 0} \frac{(-1)^r}{(2r+1)^2} = 0.29156...$$

Moreover, since the problem is reduced to evaluation of a determinant, the quantity $f(n)$ can be computed numerically for any value of n in an efficient manner. In fact, this technique is more general and allows the number of dimer coverings of any *planar* graph (or indeed of any family of graphs with fixed genus) to be computed efficiently [41].

Unfortunately, these methods do not extend to two-dimensional lattices with dimer density less than 1, or to lattices in higher dimensions even when the dimer density remains 1. In fact, the three-dimensional dimer covering problem, which asks for the number, $f(n)$, of ways of filling an $n \times n \times n$ rectangular lattice with dimers, is one of the classical unsolved problems of solid-state chemistry. A few facts are known: for example, $\ln(f(n))/n^3$ tends to a finite limit λ as n tends to infinity [21]. Hammersley [22] proved the lower bound $\lambda \geq 0.418347$, while the early paper by Fowler and Rushbrooke [17] showed the upper bound $\lambda \leq 0.54931$. It has been conjectured that λ lies between 0.43 and 0.45. In other work, Bhattacharjee et al [4] studied the phase transition behavior of the three-dimensional model. However, no reliable method is known for computing $f(n)$ to good accuracy. A similar lack of rigorous results holds for the problem at dimer densities less than 1, even in two dimensions. Notable exceptions are series expansions valid at low densities [19] and lower bounds on the free energy [6, 26].

2.1.2 Results

We make progress on the monomer-dimer problem in cases where the technique of Fisher, Kasteleyn and Temperley fails. Specifically, we give a polynomial-time algorithm for computing, to arbitrary precision, the number of coverings of a rectangular lattice in *any* dimension with *any* specified dimer density.

More precisely, for a fixed dimension d, let G be the d-dimensional rectangular lattice $[1,\ldots,n]^d$ (with periodic boundary conditions). We let $f(G,s)$ denote the number of coverings of G by s dimers and $n^d - 2s$ monomers. Our main result is a fully-polynomial randomized approximation scheme (see definition 1.3.1) for computing the function $f(G,s)$ above for rectangular lattices of any dimension d. This extends previous computational techniques in two ways. First, it enables one to compute the number of dimer coverings in lattices in three and higher dimensions. And second, it enables one to count coverings with dimer density less than 1, a problem that was not approachable by the methods of Fisher, Kasteleyn and Temperley even in two dimensions.

Our algorithm provides a feasible approach to numerical computation of quantities such as $f(n)$, the number of dimer coverings of an $n \times n \times n$ rectangular lattice in three dimensions. This is apparently the first such method whose running time provably grows only polynomially with n. We should, however, inject three caveats here. First, the running time of the algorithm, though polynomial, is not small enough to be genuinely practical; nonetheless, it is quite likely that careful honing of the algorithm and its analysis will lead to a practical method. Secondly, the algorithm provides only statistical estimates of f, rather than precise values; we stress, however, that the error bars on these estimates can be made arbitrarily small, and, in contrast to previous Monte Carlo approximation methods, are completely rigorous and require no assumptions of any kind. Thirdly, although the algorithm allows $f(n)$ to be computed efficiently for each n, we do not provide bounds on the time required to compute the asymptotics of $f(n)$ as n tends to infinity, and therefore the entropy λ. This would require, in addition, bounds on the rate of convergence of the series $\ln(f(n))/n^3$.

The above result holds for lattices with *periodic* boundary conditions (i.e., the edges of the lattice are "wrapped around" to make it toroidal). In the two-dimensional case, our method extends to lattices with *fixed* boundaries: i.e., we again get a fpras for computing the number of coverings with any specified dimer density. This result again goes beyond the technique of Fisher, Kasteleyn and Temperley for planar graphs, which holds only for dimer density 1.*

Finally, we can extend the above results to a broader class of lattices. Specifically, we get a fpras for counting coverings, with any specified dimer density, of any bipartite graph that is the Cayley graph of some finite group. This includes other commonly studied lattices such as the hexagonal lattice with periodic boundary conditions.†

2.1.3 Overview of the algorithm

The algorithms mentioned above are all based on a Monte Carlo procedure due to Jerrum and Sinclair [34] and Broder [7], for approximating the number of matchings in a graph. A *matching* in a $2m$-vertex graph $G = (V, E)$ is any subset M of the edge set E such that no two edges in M have a common endpoint. Clearly, matchings of cardinality s correspond precisely to monomer-dimer arrangements in G with s dimers and $2(m - s)$ monomers. The classical monomer-dimer problem discussed in the previous two subsections is the special case in which G is the d-dimensional rectangular lattice $[1, \ldots, n]^d$ for some d.

Let us now associate with each matching M in G a *weight* $w(M) = \mu^{|M|}$, where $|M|$ denotes the cardinality of matching M and μ is a positive real number. Recall that the *monomer-dimer partition function* (or *generating function*) of G is

$$Z(G; \mu) = \sum_M w(M) = \sum_{s=0}^{m} a_s \mu^s,$$

where the coefficient a_s is the number of matchings in G of cardinality s (see equation (1.3)). Thus, in the monomer-dimer problem on G, we are trying to compute the coefficients a_s for various values of s.

The Monte Carlo method described in [34] simulates a Metropolis-style Markov chain whose state space is the set of matchings in G (see section 1.3.1). The stationary distribution of the Markov chain is

$$\pi(M) = \frac{\mu^{|M|}}{Z(G; \mu)}.$$

The stationary probability of a matching M is proportional to its weight, and the normalizing factor is just the partition function.

Transitions correspond to local random perturbations such as the addition, deletion or exchange of an edge of the matching. More precisely, suppose we are at a state M (a matching in G). The graph H underlying the Markov chain is defined so that each state M

*If the number of monomers is some fixed constant $2c$ (so that the dimer density tends to 1 as $n \to \infty$) then the Fisher, Kasteleyn and Temperley technique can in principle be used, as follows. For each possible set of $2c$ positions for the monomers, use the technique to count dimer coverings in the graph formed by removing these sites from the lattice: this works because the graph remains planar. Now sum over all possible positions for the monomers. However, this approach no longer runs in polynomial time if c is allowed to grow with n, and is inefficient even for quite small fixed values of c.

†An analytic solution to the dimer covering problem for this planar lattice has been known for some time [65, 41]. In contrast to the rectangular lattice, the assumption of periodic boundary conditions is important here: Elser [14] has solved the dimer covering problem on a hexagonal lattice with fixed boundaries, and shown that the result depends significantly on the shape of the boundary.

has at most $|E|$ neighbors, where $|E|$ is the number of edges in G. To move to a new state, we first randomly choose an edge from the graph G. Either the edge e corresponds to a new state M' (in a manner to be described), in which case we move to M' with probability $A(M, M')$; otherwise we remain at M. When $\mu < 1$ the transitions of the Markov chain are as follows.[‡]

(i) if $e \in M$, move to the state $M' = M - e$ (i.e., $A(M, M') = 1$).

(ii) if exactly one of the vertices u or v is matched by an edge $e' \in M$, move to the state $M' = M - e' + e$ (i.e., $A(M, M') = 1$).

(iii) if neither endpoint of e is matched in M, move to the state $M' = M + e$ with probability $A(M, M') = \mu$.

(iv) in all other cases do nothing.

This Markov chain is ergodic and reversible, and from the detailed balance equation (1.4) it is easy to see that the unique stationary distribution is π. The case defined above for $\mu < 1$ is of greater physical interest, but for computation purposes it is also useful to define the Markov chain for $\mu \geq 1$. The Markov chain has the same underlying graph H, so a matching M has the same set of neighbors as the Markov chain defined above. The new transition probabilities are defined as $A(M, M') = 1/\mu$ if M' is formed by removing an edge from M, and $A(M, M') = 1$ if M' is formed by adding or exchanging an edge. Again from the detailed balance equation it follows that the stationary distribution is π.

The algorithm works by observing this process at equilibrium for various values of μ. By running a series of experiments, using suitable values of μ, we can estimate the successive ratios of coefficients a_i/a_{i-1}. If our estimates are sufficiently good, then appropriate products of these estimates would yield an estimate for each of the coefficients a_i. For the details of the algorithm, see [34].

The dominant factor in the running time of this algorithm is the number of steps that need to be simulated in order for the Markov chain to reach equilibrium for the values of μ needed in the above experiments. This quantity is analyzed rigorously in [34] and shown to be a polynomial function of the ratio $\alpha(G) = a_{m-1}/a_m$, specifically $\mathrm{O}(\alpha(G)^4|E|^2)$. (This has since been improved to $\mathrm{O}(\alpha(G)^2|V||E|)$; see [58].) The Monte Carlo procedure will therefore be efficient for graphs G in which the ratio $\alpha(G)$ is small.[§]

Matchings in G of cardinality m are called *perfect matchings*, and those of cardinality $m-1$ are called *near-perfect matchings*: these correspond respectively to dimer coverings and coverings with precisely two monomers. The ratio $\alpha(G)$ measures the amount by which the number of near-perfect matchings in G exceeds the number of perfect matchings.

[‡]The transition probabilities of the actual Markov chain described in [34] are a slight variant of these where we allow self-loops at each state to ensure aperiodicity.

[§]In [34, 59] it is shown that the same approach yields a fpras for the partition function Z — though not for all its coefficients — in an *arbitrary* graph, regardless of the value of $\alpha(G)$. Moreover, it is also possible to obtain all the coefficients a_s with $s \leq (1-\xi)m$, in time polynomial in $m^{1/(1-\xi)}$. This enables one to approximately count the number of coverings with fixed dimer density $p = 1 - \xi$, for $\xi > 0$, but the running time grows exponentially with p^{-1}.

Note that this ratio is always at least m, since the removal of any edge from a perfect matching yields a unique near-perfect matching. For an efficient algorithm, we want the ratio to be not too much larger than m: specifically, for a fpras it must be bounded above by a polynomial function of m for the family of graphs in question. Note that this is not a trivial property: it is not hard to construct a family of $2m$-vertex graphs, $m = 1, 2, \ldots$, for which the ratio grows exponentially with m. We will discuss this issue in more detail in section 2.4.

Our main technical contribution is to prove that the ratio is small for lattices and, more generally, for arbitrary Cayley graphs. Specifically, we show that if G is the d-dimensional rectangular lattice $[1, \ldots, n]^d$ with periodic boundary conditions (so that $m = \frac{1}{2}n^d$), then $\alpha(G) \leq m^2 = \frac{1}{4}n^{2d}$. This ensures that the Monte Carlo algorithm is in fact a fpras: i.e., its running time grows only polynomially with n for any fixed dimension d. A similar bound holds for arbitrary Cayley graphs. We stress that our Monte Carlo algorithm differs from earlier ones for monomer-dimer systems (see, e.g., [23]) in that it is guaranteed (independent of any heuristic arguments) to provide statistically reliable estimates in a running time that grows only polynomially with the number of lattice sites.

Our proofs of the above bound for lattices and Cayley graphs, presented in the next two sections, are elementary and rely on a novel "translation" technique: the strong symmetry properties of the lattice (and of arbitrary Cayley graphs) allow any matching (monomer-dimer configuration) to be translated, which in turn permits the symmetry to be broken. We conjecture that this technique may shed more light on other quantities related to monomer-dimer systems, and in particular the correlation between monomers at two specified sites, as studied in two dimensions by Fisher and Stephenson [16].

The remainder of the chapter is organized as follows. In the next section, we prove bounds of the above form for rectangular lattices with periodic boundary conditions in any dimension, and with fixed boundaries in two dimensions. In section 2.3 we extend our technique to handle arbitrary Cayley graphs. Finally, in section 2.4 we conclude with some further remarks on the physical and combinatorial significance of the above ratio, together with some open problems.

2.2 Rectangular Lattices

We begin by introducing some definitions and notation concerning lattices. We will be interested in two classes of lattices: the first class are those with *fixed boundary conditions*, in which the lattice is not perfectly regular but has distinguished boundary vertices. Thus, we consider the d-dimensional *rectangular* (or cartesian) lattice $L(n, d)$, where the vertices are the n^d integer lattice points in $[1, n]^d$, and two points x, y are connected by an edge iff they are unit distance apart. The second class is lattices with *periodic boundary conditions*, in which the lattice includes wrap-around edges to make it toroidal; that is, we augment $L(n, d)$ with an edge between $(x_1, \ldots, x_{i-1}, n, x_{i+1}, \ldots, x_d)$ and $(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_d)$, for each i. We will write $\widetilde{L}(n, d)$ for the periodic lattice.

We shall adopt the terminology of graphs and matchings introduced in section 2.1.3. We view $L(n, d)$ and $\widetilde{L}(n, d)$ as graphs with $2m = n^d$ vertices, and we always assume that n is even, so that both $L(n, d)$ and $\widetilde{L}(n, d)$ contain a perfect matching (dimer covering). For any $2m$-vertex graph, we let $\mathcal{M}$ be the set of perfect matchings and $\mathcal{N}$ the set of near-perfect matchings (monomer-dimer coverings with exactly two monomers). In any matching (monomer-dimer covering), we refer to the set of unmatched vertices in the graph as *holes*, and we write $\mathcal{N}(u,v)$ for the set of near-perfect matchings with holes u and v. All the graphs we consider will be bipartite, with m vertices on each side of the bipartition. It will sometimes be convenient to view the vertices on one side of the bipartition as being colored white, and those on the other side black. (In the case of the two-dimensional lattice, this coloring corresponds to the usual black and white coloring of the checker-board squares which form the dual graph.) Note that in any near-perfect matching, one hole is white and the other black.

Recall from the previous section that our aim is to construct efficient approximation algorithms for the number of monomer-dimer coverings of various lattice graphs with any specified number of dimers. As indicated earlier, all our algorithms are based on an earlier result of Jerrum and Sinclair about counting matchings in a graph subject to a certain condition on the graph. We now state this result precisely. Recall that, for a $2m$-vertex graph G, the quantity $\alpha(G) = |\mathcal{N}|/|\mathcal{M}|$ is defined to be the ratio of the number of near-perfect matchings to the number of perfect matchings in G.

Theorem 2.2.1 (Jerrum and Sinclair [34, Theorem 5.3]) *There exists a fpras for the number of perfect matchings in any family of $2m$-vertex graphs G that satisfies $\alpha(G) \le q(m)$, for a fixed polynomial q.* $\square$

The remark following this result in [34] points out that this polynomial relationship between the numbers of near-perfect and perfect matchings also allows one to construct a fpras for counting matchings of arbitrary cardinality in G.

We now proceed to prove that such a relationship does in fact hold for families of lattice graphs, and (in the next section) for more general Cayley graphs. The technique that we use in our proofs relies on the structure of the union of two matchings in a graph. Consider the subgraph C consisting of the union of the edges in two perfect matchings M_1 and M_2. If we color the edges from M_1 red and those from M_2 blue, we find that every vertex is adjacent to exactly one red edge and one blue edge, so C is the union of even-length cycles, each of which alternates colors. (Some of these cycles may be trivial, consisting of a single edge colored both red and blue.) Clearly the converse is also true, i.e., any covering of the graph with even-length cycles which alternate colors defines two perfect matchings: the set of red edges and the set of blue edges.

Similarly, suppose we have two near-perfect matchings, N_1 with holes u and v, and N_2 with holes u' and v', where u, u', v and v' are distinct vertices. Then in the subgraph C defined by the union of the red edges N_1 and the blue edges N_2, vertices u, u', v and v' all have degree one and all other vertices have degree two. So C consists of even-length alternating cycles, plus two alternating paths whose endpoints are u, u', v and v'. Moreover, either both of these paths have even length or both have odd length. See figure 2.1.

Our proofs rely on the observation that, if u' is a neighbor of u and v' is a neighbor of v, then by augmenting C with edges (u,u') and (v,v'), we can ensure that every vertex has degree two. When the graph is bipartite, the resulting subgraph must consist solely of even-length cycles, and therefore the cycle containing u and u' must also contain v and v'. By recoloring some of the edges on this new cycle, we can force it to alternate colors so that

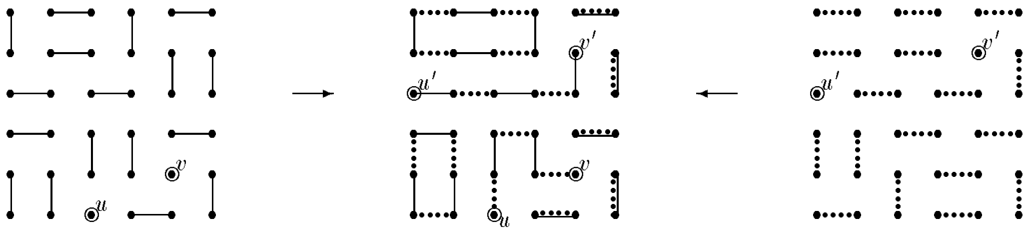


Figure 2.1: The union of two near-perfect matchings.

the cycle cover defines two perfect matchings. We use this observation to define a mapping from the set of pairs $\mathcal{N}(u,v)\times\mathcal{N}(u',v')$ to the set of pairs $\mathcal{M}\times\mathcal{M}$ that is *injective*, which in turn, by virtue of the symmetry properties of the lattice, implies that $|\mathcal{N}|$ is not much larger than $|\mathcal{M}|$.

We are now in a position to state our first result.

Theorem 2.2.2 *For the d-dimensional periodic lattice $\widetilde{L}(n,d)$, the ratio $\alpha(\widetilde{L}(n,d))$ is bounded above by $n^{2d}/4$.*

Before proving this theorem, we combine it with Theorem 2.2.1 to obtain the following immediate corollary.

Corollary 2.2.3 *There exists a fpras for the number of monomer-dimer coverings with any specified number of dimers in the d-dimensional periodic lattice $\widetilde{L}(n,d)$, for any fixed dimension d.*

Proof of Theorem 2.2.2. Let $\mathcal{M}$ and $\mathcal{N}$ be the sets of perfect and near-perfect matchings respectively in $\widetilde{L}(n,d)$, so that $\alpha(\widetilde{L}(n,d)) = |\mathcal{N}|/|\mathcal{M}|$. First we fix two holes, u and v. Let u' be the neighbor one to the right of u, i.e., $u' = u + (1,0,...,0) \bmod n$. Similarly, let v' be the neighbor one to the right of v.

We proceed to construct an injection ϕ from $\mathcal{N}(u,v)\times\mathcal{N}(u',v')$ into $\mathcal{M}\times\mathcal{M}$. To do this, let $N_1\in\mathcal{N}(u,v)$ and $N_2\in\mathcal{N}(u',v')$, and consider the subgraph C of $\widetilde{L}(n,d)$ defined by the union of red edges N_1, blue edges N_2 and *special edges* (u,u') and (v,v'). If we color the special edges red, then u' and v' are each adjacent to two red edges, and every other vertex is adjacent to one edge of each color; if we now flip the colors of the edges along one of the paths from u' to v', every vertex will be adjacent to exactly one edge of each color. To avoid ambiguity, we choose the path from u' to v' which does not pass through u. As we saw earlier, the sets of colored edges now define two perfect matchings. See figure 2.2.

We need to check that this map ϕ is injective: given any pair of perfect matchings (M_1,M_2) in the image of the map, we show that we can uniquely reconstruct the pair of near-perfect matchings, one with holes u and v and the other with holes u' and v', that are mapped by ϕ to (M_1,M_2). Note that the union of any pair of matchings in the image of ϕ always contains an alternating cycle that includes the edges (u,u') and (v,v'). Now

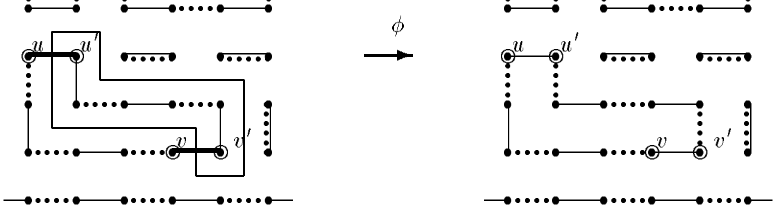


Figure 2.2: Mapping two near-perfect matchings to two perfect matchings.

color the edges of the matching containing (u, u') red, and the edges of the other matching blue. By flipping the colors of the edges along the path from u' to v' (again choosing the path which avoids u, for consistency), we make u' adjacent to two red edges. Since u' and v' are the holes of some near-perfect matching, they lie on opposite sides of the bipartition and any path between them must have odd length. Therefore, after the flipping operation v' must be adjacent to two red edges as well, while all other vertices are still adjacent to one edge of each color. If we now remove the edges (u, u') and (v, v'), the colored edges must correspond to the two near-perfect matchings that are mapped by ϕ to (M_1, M_2).

The above construction demonstrates that $|\mathcal{N}(u,v)| \cdot |\mathcal{N}(u',v')| \leq |\mathcal{M}|^2$. To finish the proof, we use the structure of the lattice $\widetilde{L}(n,d)$: in a periodic lattice, the operation of shifting a matching one position to the right is a bijection between the sets $\mathcal{N}(u,v)$ and $\mathcal{N}(u',v')$, so $|\mathcal{N}(u,v)| = |\mathcal{N}(u',v')|$. Thus the above relationship gives $|\mathcal{N}(u,v)|^2 \leq |\mathcal{M}|^2$, which implies $|\mathcal{N}(u,v)| \leq |\mathcal{M}|$. Summing over all choices of a black vertex u and a white vertex v, we find that $|\mathcal{N}| \leq n^{2d}|\mathcal{M}|/4$, so that $\alpha(\widetilde{L}(n,d)) \leq n^{2d}/4$, as claimed. □

Remark. It should be clear from the above proof that Theorem 2.2.2 (and hence Corollary 2.2.3) generalizes to "hybrid" lattices that have fixed boundary conditions in some dimensions provided there exists at least one dimension in which the lattice has periodic boundary conditions. It also holds in more general bipartite rectangular lattices of size $n_1 \times n_2 \times \ldots \times n_d$ with periodic boundary conditions (i.e., for any dimension i in which the boundary is periodic, n_i must be even). □

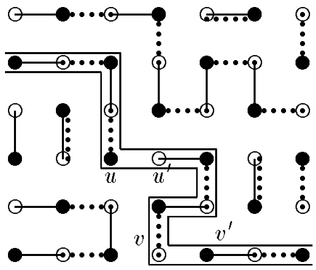
The following theorem extends the above technique to handle two-dimensional lattices with *fixed* boundaries. We again show that in these lattices the number of near-perfect matchings cannot be too large compared to the number of perfect matchings, and then appeal to Theorem 2.2.1.

Theorem 2.2.4 *For the two-dimensional lattice with fixed boundaries $L(n,2)$, the ratio $\alpha(\widetilde{L}(n,d))$ is bounded above by $n^4/4$.*

Corollary 2.2.5 *There exists a fpras for the number of monomer-dimer coverings with any specified number of dimers in the two-dimensional lattice with fixed boundaries $L(n,2)$.* □

Proof of Theorem 2.2.4. We will prove the theorem for the slightly more general case of $n_1 \times n_2$ lattices with fixed boundaries, where n_1 is even. Let τ be a map which shifts the lattice $L(n,2)$ one position to the right in $\mathbb{Z}^2$; that is, for a vertex $w = (w_1, w_2)$, define $\tau(w) = (w_1 + 1, w_2)$. This map extends to matchings in the natural way: if N is a matching, then $\tau(N) \in [2, n_1 + 1] \times [1, n_2]$ is just: $(\tau(x), \tau(y)) \in \tau(N)$ iff $(x, y) \in N$.

Let $\mathcal{M}$ and $\mathcal{N}$ be the sets of perfect and near-perfect matchings respectively in the lattice $L(n,2)$. As in the last proof, we will fix holes u and v and show that $|\mathcal{N}(u,v)| \leq |\mathcal{M}|$. We again define an injection $\phi : \mathcal{N}(u,v) \times \mathcal{N}(u,v) \hookrightarrow \mathcal{M} \times \mathcal{M}$ as follows. Let $N_1, N_2 \in \mathcal{N}(u,v)$ be two near-perfect matchings. Consider the subgraph C obtained by taking the union of N_1 with a shifted version of N_2 and adding the two special edges as before, i.e., $C = N_1 \cup \tau(N_2) \cup \{(u,u'), (v,v')\}$, where $u' = \tau(u)$ and $v' = \tau(v)$. Then all the vertices in the leftmost column 1 and the rightmost column $n+1$ have degree one in C, and all other vertices have degree two. Thus C is the union of even-length cycles and paths with each endpoint in either the first or $(n+1)$st column (see figure 2.3). Color the edges from N_1 red and the edges from N_2 blue.



1 $\quad n \quad n+1$

Figure 2.3: Union of N_1 and $\tau(N_2)$.

We will argue that, because the two-dimensional lattice is planar, any path or cycle which passes through u and u' must also pass through v and v'. This is immediate if u and u' lie on a cycle, so we focus on the case where u and u' lie on a path. The proof is by contradiction, and there are two cases to consider (see figure 2.4).

First, suppose that we have a path P from the first column to the $(n+1)$st column which passes through u and u', and not through v and v'. Without loss of generality we can assume that v and v' lie below P. Then P starts with a red edge, ends with a blue edge, and has one special edge, so it has odd length. It follows that if P starts at a black vertex then it ends at a white vertex, and conversely. Therefore, the number of vertices in the first column above P has opposite parity to the number of vertices in the $(n+1)$st column above P. (Since n is even, corresponding vertices in each of these columns fall on the same side of the black-and-white bipartition.) But consider the set of all vertices that lie above the path P. There must be an even number of these vertices lying in the first through nth columns, since these vertices are matched in N_1, and an even number lying in the second through $(n+1)$st columns, since these vertices are matched in N_2. This is a contradiction.

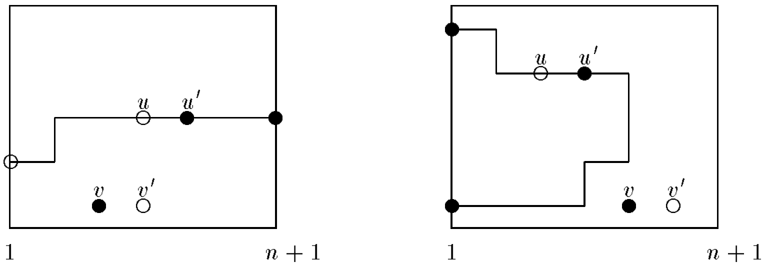


Figure 2.4: Proof of Theorem 2.2.4.

Second, suppose that P, the path going through u and u', starts and ends in the first column. By interchanging the roles of u, u' and v, v' if necessary, we may assume without loss of generality that v and v' lie outside the cycle defined by the path P and the first column. Now P starts and ends with a red edge and has one special edge, so it must have even length. If it starts at a black vertex it ends at a black vertex, and conversely, so there are an odd number of vertices in the first column that lie between these endpoints. Let S be the set of points that lie strictly inside the path P. Then $|S|$ must be even since N_1 matches all the vertices in S. But N_2 matches all the vertices in S except those which lie in the first column, a contradiction since this number is odd.

Therefore we can conclude that u, u', v, v' all lie on the same even-length cycle or the same path. In either case we can proceed as in the proof of Theorem 2.2.2: color the special edges red and then flip the colors of the edges along the path between u' and v' (in the case of a cycle, where this is ambiguous, we always choose the path which does not pass through u). The sets of colored edges then define two perfect matchings M_1 and $\tau(M_2)$.

Furthermore, given any two matchings in the image of the map ϕ we can uniquely reconstruct the pair of near-perfect matchings which are their preimage, so ϕ is injective. To see this, note that any element in the image of ϕ consists of two perfect matchings M_1 and M_2 such that $M_1 \cup \tau(M_2)$ contains a cycle or path which passes through all of u, u', v, v', and from here we can reconstruct N_1 with holes u and v and $\tau(N_2)$ with holes u' and v' by reversing the color flipping operation as in the proof of Theorem 2.2.2. Thus we have $|\mathcal{N}(u,v)| \leq |\mathcal{M}|$. Summing over choices of u and v, we get $|\mathcal{N}| \leq n_1^2 n_2^2 |\mathcal{M}|/4$, which yields the required bound on $\alpha(G)$. $\square$

2.3 Other Lattices

2.3.1 Bipartite Cayley graphs

The following theorem extends the techniques from the last section to handle other lattices. More precisely, we can, in polynomial time, approximately count the number of monomer-dimer coverings with any specified number of dimers in any $2m$-vertex bipartite graph which is the Cayley graph of some finite group.

Recall that the *Cayley graph* of a group G with a given set of generators is defined by identifying vertices with words in G and connecting vertices x and y by an edge in the graph if, for some generator a of G, $xa = y$. This class of graphs includes any finite hexagonal lattice which has periodic boundaries around some fundamental domain. One group which generates this lattice is $\langle a, b, c \mid a^2, b^2, c^2, (abc)^2, (ab)^i, (bc)^i, (ca)^i \rangle$, for any integer i, where a, b and c are the generators and the words which follow are relators equivalent to the identity in the group. See figure 2.5.

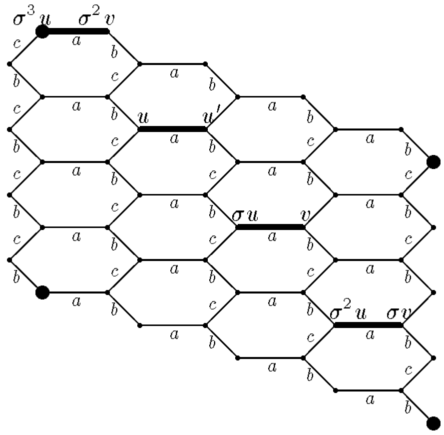


Figure 2.5: Proof of Theorem 2.3.1 for the hexagonal lattice
$\langle a, b, c \mid a^2, b^2, c^2, (abc)^2, (ab)^4, (bc)^4, (ca)^4 \rangle$

Theorem 2.3.1 *Let G be a $2m$-vertex bipartite graph which is the Cayley graph of a finite group. Then $\alpha(G) \leq m^2$.*

Note that this theorem, applied to the Cayley graph $\widetilde{L}(n, d)$, yields precisely the same bound on α as Theorem 2.2.2.

Corollary 2.3.2 *There exists a fpras for the number of monomer-dimer coverings with any specified number of dimers in any bipartite graph which is the Cayley graph of a finite group.*
□

Proof of Theorem 2.3.1. Given a group G, we consider its Cayley graph, which we assume to be bipartite. Choose a vertex and label it with e, the identity element of G. This determines a label in G for every vertex in the graph, corresponding to the product of the labels along any path leading to it from the identity vertex.

Fix a pair of holes u and v in the graph. Let $u' = ua$ be the neighbor of u defined by some fixed generator a. Let $\sigma = vu'^{-1}$ be the word in G which maps u' to v by multiplication on the left. Then, since group multiplication on the left preserves adjacency in G, σu is a neighbor of v. Moreover, since the group is finite, there exists some k such that $\sigma^k = e$. We will show that $|\mathcal{N}(u,v)| \le |\mathcal{M}|$ by exhibiting an injection ϕ from the cartesian product $\prod_{i=1}^{k} [\mathcal{N}(\sigma^{i-1}u, \sigma^{i-1}v)]$ into $\mathcal{M}^k$.

We define the map ϕ in $k-1$ stages. Let $N_i \in \mathcal{N}(\sigma^{i-1}u, \sigma^{i-1}v)$, for $1 \le i \le k$, be a set of near-perfect matchings. Stage one maps the pair (N_1, N_2) into (M_1, N_2'), where M_1 is a perfect matching and N_2' is an "auxiliary" near-perfect matching. In stage j, for $2 \le j \le k-2$, we map the pair (N_j', N_{j+1}) into (M_j, N_{j+1}'), where N_j' is the auxiliary near-perfect matching from the previous stage. The $(k-1)$st stage maps (N_{k-1}', N_k) into the final pair of perfect matchings (M_{k-1}, M_k).

In the first stage we consider the subgraph $C_1 = N_1 \cup N_2 \cup \{(\sigma u, v)\}$. We color the edges from $N_1 \cup \{(\sigma u, v)\}$ red and those from N_2 blue. Then all vertices have degree two except u and σv, each of which has degree one, and v is the only vertex that has two edges of the same color incident to it. By flipping the colors of the edges along the portion of the path from v to σv, we can force the path from u to σv to have alternating colors. Because the graph is bipartite, the two vertices of degree one, u and σv, will both be adjacent to a blue edge. Then the blue edges form a perfect matching, M_1, and the red edges form the first auxiliary near-perfect matching, N_2', with holes u and σv.

Beginning the jth stage in this mapping, we have already mapped $\prod_{i=1}^{k} [\mathcal{N}(\sigma^{i-1}u, \sigma^{i-1}v)]$ into $\mathcal{M}^{j-1} \times \mathcal{N}(u, \sigma^{j-1}v) \times \prod_{i=j+1}^{k} [\mathcal{N}(\sigma^{i-1}u, \sigma^{i-1}v)]$. Stage j itself will consist of an injection from $\mathcal{N}(u, \sigma^{j-1}v) \times \mathcal{N}(\sigma^j u, \sigma^j v)$ into $\mathcal{M} \times \mathcal{N}(u, \sigma^j v)$. In particular, we will map the pair (N_j', N_{j+1}) to (M_j, N_{j+1}'), as follows.

If we consider the subgraph $C_j = N_j' \cup N_{j+1} \cup \{(\sigma^j u, \sigma^{j-1}v)\}$, we get an odd-length path from u to $\sigma^j v$. By flipping the colors of the edges along the portion of the path from $\sigma^{j-1}v$ to $\sigma^j v$, we get a perfect matching, M_j, and a near-perfect matching, N_{j+1}', with holes u and $\sigma^j v$.

At the $(k-1)$st stage the mapping terminates. Here, we are mapping $N_{k-1}' \in \mathcal{N}(u, \sigma^{k-2}v)$ and $N_k \in \mathcal{N}(\sigma^{k-1}u, \sigma^{k-1}v)$. But $v = \sigma u'$, so $\sigma^{k-1}v = \sigma^k u' = u'$, since σ^k is the group identity. Therefore, the subgraph $N_{k-1}' \cup N_k \cup \{(\sigma^{k-1}u, \sigma^{k-2}v), (u, u')\}$ consists only of even-length cycles. By flipping colors along one of the paths from $\sigma^{k-2}v$ to u' (choosing the path which passes through u, to avoid ambiguity), we get even cycles with alternating colors: again this follows because the Cayley graph is bipartite. The two sets of colored edges now define the final two perfect matchings M_{k-1} and M_k.

Given the labels of the holes u and v, the vertex $u' = ua$ is uniquely determined, as is the word $\sigma = vu'^{-1}$ and its inverse. We can then invert the map ϕ by working backwards in stages, each stage being similar to the proof of Theorem 2.2.2. This shows that ϕ is injective, and therefore $\prod_{i=1}^{k} |\mathcal{N}(\sigma^{i-1}u, \sigma^{i-1}v)| \le |\mathcal{M}|^k$.

The last step is to see that, for any word σ, translation by σ^i is a bijection between matchings $\mathcal{N}(u,v)$ and matchings $\mathcal{N}(\sigma^i u, \sigma^i v)$. More precisely, we extend σ^i to matchings by defining $(\sigma^i x, \sigma^i y) \in \sigma^i(N)$ iff $(x,y) \in N$, where $N \in \mathcal{N}(u,v)$. This is valid since if x and y are neighbors in the Cayley graph then there is some generator b such that $y = xb$, so $\sigma^i x$ and $\sigma^i y = \sigma^i xb$ are also neighbors. And, since u and v are unmatched in N, $\sigma^i u$ and $\sigma^i v$ are the unmatched vertices in $\sigma^i(N)$, so $\sigma^i(N) \in \mathcal{N}(\sigma^i u, \sigma^i v)$. Thus $|\mathcal{N}(u,v)| = |\mathcal{N}(\sigma^i u, \sigma^i v)|$ for any i. Combining this with the fact that ϕ is injective, we see that $|\mathcal{N}(u,v)|^k \le |\mathcal{M}|^k$, and hence $|\mathcal{N}(u,v)| \le |\mathcal{M}|$. Since G has $2m$ vertices, summing over u and v gives $|\mathcal{N}| \le m^2|\mathcal{M}|$ as claimed. $\square$

The proof of theorem 2.3.1 actually holds for a larger class of graphs which we call *neighbor-automorphism* graphs. A graph G is in this class if for every pair of vertices u and v, there exists an automorphism ϕ from G onto itself such that $\phi(u)$ is a neighbor of v. Notice that this class includes Cayley graphs, since they are vertex transitive. However, it includes a much larger set of graphs which need not even be regular. The proof of theorem 2.3.1 demonstrates that for any bipartite neighbor-automorphism graph G with vertex sets V_1 and V_2, the ratio $\alpha(G) < |V_1||V_2|$.

This is noteworthy when coupled with the following theorem.

Theorem 2.3.3 *Counting the number of perfect matchings in a bipartite neighbor-automorphism graph is #P-complete.*

The proof uses a reduction from counting perfect matchings in an arbitrary bipartite graph; this was also shown by Valiant to be #P-complete [64]. Briefly, the idea behind the reduction is as follows. Let $G = (V_1, V_2, E)$ be an arbitrary bipartite graph. We will construct a new bipartite graph $H_k = ((V_{11} \cup V_{12}), (V_{21} \cup V_{22}), E')$, where the subgraph induced on $V_{11} \cup V_{21}$ is isomorphic to G, as is the subgraph induced on $V_{12} \cup V_{22}$. The edge set connecting V_{11} and V_{22} is a multigraph consisting of k copies of the complete graph. Similarly, k copies of the complete graph connect V_{12} and V_{21}. It is easy to verify that H_k is a neighbor-automorphism graph. For any fixed value of k, the number of perfect matchings in H_k is a linear combination of $a_1^2, a_2^2, ..., a_m^2$, where a_i is the number of matchings of size i in G. Thus, if we could determine the number of perfect matchings in each H_j, for $1 \leq j \leq m$, we would be able to solve a system of linear equations to determine the value of each a_i, including a_m, the number of perfect matchings in G.

2.3.2 Non-bipartite Cayley graphs

Recently Jerrum pointed out that the ratio $\alpha(G)$ of near-perfect matchings and perfect matchings could be bounded for any Cayley graph using an argument based on the techniques presented in the previous two sections. This gives us a fpras for important non-bipartite periodic lattices, such as the triangular lattice and the face- and body-centered cubic lattices. We include this argument for completeness.

Theorem 2.3.4 *[Jerrum] Let G be a $2m$ vertex graph which is the Cayley graph of a finite group. Then $\alpha(G) \leq m^3$.*

This, of course, gives us the immediate corollary:

Corollary 2.3.5 *There exists a fpras for the number of monomer-dimer coverings with any specified number of dimers in any graph which is the Cayley graph of a finite group.* □

Proof of Theorem 2.3.4. Recall that for any Cayley graph G, the vertices are labeled with words in the group. We will consider all near-perfect matchings $\mathcal{N}(u, v)$ with fixed holes u and v in the graph and will show that $|\mathcal{N}(u, v)| \leq 2m|\mathcal{M}|$. Summing over all choices of pairs u and v, this will give us a m^3 bound for the ratio $\alpha(G)$.

For a fixed pair of vertices u and v in the Cayley graph, we define the distance d to be the length of a shortest path between u and v. Clearly $d \leq 2m$ since the graph has $2m$ vertices. We will show that $|\mathcal{N}(u,v)| \leq d|\mathcal{M}|$ by induction on d.

Suppose $d = 1$. Then u and v are neighbors in the Cayley graph. We can injectively map the set of near-perfect matching with holes u and v into the set of perfect matchings by adding the edge (u,v). Therefore $|\mathcal{N}(u,v)| \leq |\mathcal{M}|$ when u and v are neighbors.

Now assume inductively that we have verified the claim for all $d < \hat{d}$. Let u and v be two vertices at distance $\hat{d}$ in the Cayley graph. Let v_0 be the neighbor of v on a shortest path between u and v. Consider the set of near-perfect matchings where one of the holes is v_0 and let u_0 be any vertex so that the number of near-perfect matchings with holes v_0 and u_0 is maximized (i.e., $|\mathcal{N}(u_0,v_0)| \geq |\mathcal{N}(u',v_0)|$ for all $u' \in G$). Note that because the graph is vertex transitive, $|\mathcal{N}(u_0,v_0)| \geq |\mathcal{N}(u',v')|$ for all choices of u' and v'.

We define a map ϕ on pairs of near-perfect matchings $\mathcal{N}(u,v) \times \mathcal{N}(u_0,v_0)$. Choose $N_1 \in \mathcal{N}(u,v)$ and $N_2 \in \mathcal{N}(u_0,v_0)$. Color the edges of N_1 red and those of N_2 blue. Consider the union of edges $N_1 \cup N_2 \cup \{(v,v_0)\}$. Now every vertex except for u and u_0 has degree 2. There are two cases: either we have formed a path between u and u_0 and a cycle including v and v_0, or we have formed a single path between u and u_0.

In the first case we can flip the colors on the cycle and remove the edge (v,v_0). Now the red edges form a near-perfect matching with holes u and v_0 and the blue edges form a near-perfect matching with holes v and u_0. In the second case we can color the edge (v,v_0) red and flip the colors on the part of the path from v to u_0. The red edges form a near-perfect matching with holes u and u_0 and the blue edges form a perfect matching. In either case we can uniquely reconstruct the original near-perfect matchings from the image of the map. Summarizing this discussion, we find ϕ is an injective map between the following sets:

$$\phi : \mathcal{N}(u,v) \times \mathcal{N}(u_0,v_0) \hookrightarrow (\mathcal{N}(u,v_0) \times \mathcal{N}(u_0,v)) \cup (\mathcal{N}(u,u_0) \times \mathcal{M}).$$

It follows that

$$|\mathcal{N}(u,v)| \cdot |\mathcal{N}(u_0,v_0)| \leq |\mathcal{N}(u,v_0)| \cdot |\mathcal{N}(u_0,v)| + |\mathcal{N}(u,u_0)| \cdot |\mathcal{M}| \tag{2.1}$$

Recall that $|\mathcal{N}(u_0,v_0)|$ was chosen so as to be maximal over all choices of holes u_0 and v_0. In particular $|\mathcal{N}(u_0,v_0)| \geq |\mathcal{N}(u_0,v)|$ and $|\mathcal{N}(u_0,v_0)| \geq |\mathcal{N}(u,u_0)|$. Combining this with equation (2.1) we find

$$|\mathcal{N}(u,v)| \cdot |\mathcal{N}(u_0,v_0)| \leq |\mathcal{N}(u,v_0)| \cdot |\mathcal{N}(u_0,v_0)| + |\mathcal{N}(u_0,v_0)| \cdot |\mathcal{M}|$$

and hence

$$|\mathcal{N}(u,v)| \leq |\mathcal{N}(u,v_0)| + |\mathcal{M}|. \tag{2.2}$$

Since the distance between u and v_0 is $\hat{d} - 1$, we know inductively that

$$|\mathcal{N}(u,v_0)| \leq (\hat{d} - 1)|\mathcal{M}|.$$

Combining this with equation (2.2) we find

$$|\mathcal{N}(u,v)| \leq \hat{d}|\mathcal{M}|,$$

thereby completing the induction.

Hence, for any choice of u and v, $|\mathcal{N}(u,v)| \leq 2m|\mathcal{M}|$. Summing over all $m^2/2$ distinct pairs of vertices u and v, we can conclude that $|\mathcal{N}| \leq m^3|\mathcal{M}|$. $\square$

Note that this gives an alternative proof of theorem 2.3.1. For a bipartite graph only one of the two cases described in the proof is possible, and equation (2.2) becomes

$$|N(u,v)| \leq |M|.$$

This proves the bound $\alpha(G) \leq m^2$ for bipartite Cayley graphs.

2.4 Concluding Remarks and Open Problems

We have used elementary combinatorial techniques to show that, for any Cayley graph G, the quantity $\alpha(G)$ is small, i.e., the number of near-perfect matchings (monomer-dimer coverings with two monomers) exceeds the number of perfect matchings (dimer coverings) in G by only a small polynomial factor. This allowed us to deduce rigorous polynomial time bounds for a Monte Carlo algorithm for counting coverings in such graphs with any specified number of dimers.

Our results show that, for a $2m$-vertex bipartite Cayley graph G, the quantity $\alpha(G)$ lies in the range $[m, \frac{1}{4}m^2]$. (The upper bound comes from Theorem 2.3.1, while the lower bound is trivial—see section 2.1.3.) It would be interesting to know whether either of these bounds can be improved for Cayley graphs, and to determine the precise value of α for the d-dimensional rectangular lattice $\widetilde{L}(n,d)$. Similarly, for non-bipartite lattices we only know the weaker bound $\alpha(G) \leq m^3$. Apart from their inherent interest, improving these bounds would affect the efficiency of the Monte Carlo algorithm since the quantity $\alpha(G)$ enters into the running time as explained in section 2.1.3.

Our technique also breaks down in the case of lattices with fixed boundary conditions (in dimensions higher than two). Techniques similar to those we have presented can be used to reduce the question of bounding α to that of establishing the *local* property that the number of near-perfect matchings with fixed holes u and v is polynomially related to the number of matchings with holes u' and v', where u' is a neighbor of u and v' is a neighbor of v. However, we have been unable to use this observation to obtain a proof for fixed boundary conditions in higher dimensions.

One can go further and ask for a characterization of those families of graphs for which the ratio α is polynomially bounded, and hence for which the monomer-dimer problem is tractable using the above Monte Carlo approach. This question is also of considerable combinatorial interest, since counting perfect matchings (dimer coverings) in a bipartite graph is equivalent to computing the *permanent* of a 0-1 matrix [51]. This is a widely studied problem in combinatorics for which the existence of an efficient approximation algorithm is an important open question in the theory of computation [64]. The Monte Carlo algorithm sketched above runs in polynomial time for a wider class of graphs than any other currently known algorithm, so it is of interest to establish precisely which graphs are amenable to it. (For other simpler, but apparently less widely applicable approximation algorithms, see [38, 33] and [55].) Moreover, it is conceivable that any graph G for which $\alpha(G)$ is large can be efficiently decomposed in such a way that the resulting components have a small value of α, and hence fall within the scope of the Monte Carlo algorithm; this idea was used in [37] to obtain an approximation scheme for general graphs whose running time, though still exponential, improves substantially on naïve deterministic methods.

The question of whether α is polynomially bounded for a given family of graphs is apparently rather subtle. It is not hard to construct "bad" examples. Consider, for example, the family of graphs $\{G_n : n = 1, 2, \ldots\}$ defined in figure 2.6, where G_n has $2m = 4n + 2$ vertices. It is easy to see that G_n has only one perfect matching but more than $2^n = 2^{(m-1)/2}$ near-perfect matchings (consider just those with holes at u and v), so the ratio $\alpha(G_n) > 2^n$ is exponentially large. On the other hand, α is known to be polynomially bounded for all sufficiently dense graphs, all graphs with sufficiently good "expansion" properties, and almost every random graph in a suitable model [34]. Interestingly, the technique used to prove this property in all these cases is not applicable to lattices since it involves constructing short augmenting paths for near-perfect matchings; such paths do not exist in lattice graphs, which have large diameter. The injective mapping technique presented here is therefore a new approach, and we hope that it will lead to a better understanding of the behavior of the ratio α in general graphs.

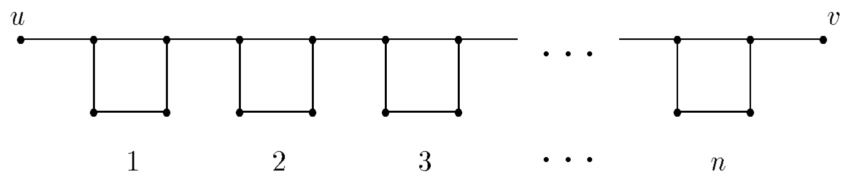


Figure 2.6: The "bad" graph G_n

In the case of lattices, we conjecture that the explicit mappings we have exhibited between near-perfect matchings with two fixed holes and perfect matchings might shed light on the behavior of the number of near-perfect matchings as a function of the positions of the holes. In physical terms, this corresponds to the correlation between a pair of monomers in a sea of dimers, a quantity for which partial results were obtained in two dimensions by Fisher and Stephenson [16]. Our techniques immediately yield a simple and rigorous proof that, in any dimension d, the number of configurations with two monomers at any fixed pair of vertices u, v is bounded by n^{-d} times the number of configurations with two adjacent monomers. A more careful analysis may enable one to make more precise statements about this correlation.

Chapter 3

Self-avoiding walks

3.1 Introduction

3.1.1 Background

Self-avoiding walks in $\mathbb{Z}^d$ have been studied by mathematicians and natural scientists for many years and are the subject of an extensive literature; for a state-of-the-art survey, see the recent book of Madras and Slade [48]. (See also the book by Lawler [45] for related topics.) One of the most important applications is as a model for the spatial arrangement of linear polymer molecules in chemical physics. Here the walk represents a molecule composed of many (perhaps 10^5 or more) monomers linked in a chain, and the self-avoidance constraint reflects the fact that no two monomers may occupy the same position in space. The *length* $|w|$ of a self-avoiding walk w is the number of edges in w. For any fixed dimension d, let $\mathcal{S}_n$ denote the set of self-avoiding walks of length n in $\mathbb{Z}^d$, and let $c_n = |\mathcal{S}_n|$ be the number of walks of length n. The two most fundamental computational problems concerning self-avoiding walks are:

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

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

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

One key fact that holds in all dimensions was discovered in 1954 by Hammersley and Morton [27]; they observed that $\lim_{n\to\infty} c_n^{1/n} = \mu$ exists, and that $\mu^n \leq c_n = \mu^n f(n)$, where $\lim_{n\to\infty} f(n)^{1/n} = 1$. This is a straightforward consequence of the obvious fact that the sequence $\ell_n = \log c_n$ is *subadditive*, i.e., $\ell_{n+m} \le \ell_n + \ell_m$ for all n, m. Hammersley and Welsh [28] later showed that $f(n) = \mathrm{O}(a^{n^{1/2}})$ for some constant a. It is a celebrated and long-standing conjecture that $f(n)$ is in fact polynomially bounded. More precisely, we have:

Conjecture 1:

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

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

Here μ, A and γ are all dimension-dependent constants. Note that the dominant behavior of c_n is the exponential function μ^n; comparing this with the case of simple walks, whose number is precisely $(2d)^n$, we see that the effect of the self-avoidance constraint is to reduce the effective number of choices the walk has at each step from $2d$ to μ. The dimension-dependent number μ is known as the *connective constant.* This crude behavior is modified by the correction term $f(n)$ as in conjecture 1. Here γ is a so-called *critical exponent.* (Note, however, that γ, unlike μ, is not even known to exist.)

Although unproven, conjecture 1 is supported by extensive (though non-rigorous) empirical studies and ingenious heuristic arguments, which have also been employed to obtain numerical estimates for the constants μ and γ. Elementary considerations show that $\mu \in (d, 2d-1)$. For $d = 2$, it has actually been proven that $\mu \in (2.62, 2.70)$ [2, 10]. (See also [29] for similar bounds in higher dimensions.) However, these rigorous bounds are much weaker than the non-rigorous estimates obtained by empirical methods, which are typically quoted to four decimal places. There are even precise conjectured values for the critical exponent γ in two and three dimensions (despite the fact that γ is not known to exist): for $d = 2$, γ is believed to be $\frac{43}{32}$, and for $d = 3$ it is believed to be approximately 1.16. (See [48] for a detailed summary of numerical estimates.)

Much effort has been invested in obtaining statistical estimates of the above quantities using Monte Carlo simulations. However, the error bars on these estimates are only justified heuristically. In this chapter, we attempt to put such experiments on a firmer footing. We present Monte Carlo algorithms for approximating the number of self-avoiding walks of a given length for a given dimension d, and for generating self-avoiding walks of a given length almost uniformly at random. The running time of our algorithms is polynomial in the walk length n and grows only slowly with parameters controlling the accuracy and confidence levels of the estimates. These are the first polynomial time algorithms where the statistical errors are rigorously controlled. Our algorithms are based on modifications and extensions of a Monte Carlo approach studied originally by Berretti and Sokal [3]. In the next subsection we sketch this approach and point out its limitations. Then, in section 3.1.3, we summarize our algorithms and explain how they overcome these problems.

3.1.2 Monte Carlo simulations

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

The above type of Markov chain was considered by Berretti and Sokal [3], who used a single parameter $\beta \leq 1$ to control the relative probabilities of extending or contracting the walk by one edge. Given a walk of length i, one of the $2d$ lattice edges incident to the free endpoint of the walk is chosen with equal probability. If the edge extends the walk so as to be self-avoiding, then it is added with probability β; if the edge is the last edge of the walk, then it is removed; otherwise, nothing is done.* Assuming conjecture 1, Berretti and Sokal argue that, for any given value of n, taking β sufficiently close to (but smaller than) μ^{-1}, where μ is the connective constant, ensures that the stationary distribution assigns reasonably high weight (i.e., $1/q(n)$ for some polynomial q) to $\mathcal{S}_n$. Furthermore, again assuming conjecture 1, Sokal and Thomas [62] argue that with such values of β the Markov chain is rapidly mixing, i.e., it gets very close to stationarity after a number of steps that is only polynomial in n (see also [46]). In order to appreciate the role of β here, consider a truncated version of this Markov chain in which the length of a walk is never allowed to exceed n, so that the stationary distribution is always well defined; if β is too much smaller than μ^{-1} then we will only generate short walks, while if β is too much larger then the Markov chain will not backtrack often enough and consequently will take a long time to reach stationarity. Thus β must be very carefully chosen. Berretti and Sokal perform their experiments by "fine-tuning" β and observing the Markov chain until the observations suggest that β is sufficiently close to μ^{-1}.

Berretti and Sokal's algorithm suffers from two drawbacks. First, one must assume conjecture 1 (for appropriate values of the constants μ, γ and A) in order to bound the time required before the Markov chain reaches stationarity. As long as conjecture 1 remains open for any choices of these constants, there is no guarantee that the algorithm produces reliable answers in polynomial time. Second, in order to implement the algorithm it is necessary to have a good estimate of μ already, since β needs to be taken a little smaller than μ^{-1}. This leads to circularity, since determining μ is one of the principal goals of the algorithm. While many similar Monte Carlo algorithms have been used to study self-avoiding walks (see Chapter 9 of [48] for a summary), all of these suffer from a similar lack of rigorous justification, and thus offer no guarantee that their results are reliable.

---

*Actually, these transition probabilities are a slightly simplified version of those used in [3], but this difference is inessential to the behavior of the chain.

3.1.3 Our results

We develop a Monte Carlo algorithm for self-avoiding walks by modifying the Markov chain used by Berretti and Sokal so as to overcome the difficulties discussed in the last subsection. We make two elementary but important innovations. First, we allow the parameter β to vary at each level of the Markov chain (i.e., we let β depend on the length of the walks), and we *calculate* an appropriate value of β at each level based on observations of the Markov chain at earlier levels. Thus we require no prior knowledge of β. Second, we show that, while the efficiency of our Markov chain simulation is still based on an assumption (conjecture 2, defined below), this is weaker than conjecture 1 and, more importantly, is tested in advance by the algorithm in the region in which it is being assumed. Thus when we run our algorithm, *either* we will gather strong evidence (in the form of a counter-example) that the conjecture is false, *or* we will know that we can trust our simulations. This notion of *self-testing*, which either gives us confidence in our results or warns us that they may be erroneous, has been previously explored in the context of program checking (see, e.g., [5]).

The conjecture we require is the following:

Conjecture 2:

For any dimension d, for some fixed polynomial g,

$$c_n c_m \leq g(n+m)\, c_{n+m}, \quad \forall n, m.$$

This conjecture says that if we choose random self-avoiding walks of lengths n and m, then with non-negligible probability we can glue the walks together to produce a self-avoiding walk of length $n+m$. To be more precise, for self-avoiding walks w_1 and w_2, define the *concatenation* $w_1 \circ w_2$ to be the walk formed by the union of the edges in w_1 and $\tau(w_2)$, where τ translates w_2 so that its origin coincides with the free endpoint of w_1. Note that $w_1 \circ w_2$ need not be self-avoiding. If w_1 and w_2 are selected uniformly at random from $\mathcal{S}_n$ and $\mathcal{S}_m$ respectively, then the quotient $\frac{C_{n+m}}{C_n C_m}$ represents the probability that $w_1 \circ w_2$ *is* self-avoiding (see figure 3.1). Conjecture 2 asserts that $g(n)^{-1}$ is a lower bound on this probability.

This conjecture is no more restrictive than conjecture 1, on which previous Monte Carlo methods, including that of Berretti and Sokal, rely. To see this, note that $c_n \sim A\mu^n n^{\gamma-1}$ implies $\frac{c_n c_m}{c_{n+m}} \sim A(\frac{nm}{n+m})^{\gamma-1} \leq A\,(\frac{n+m}{4})^{\gamma-1}$. Thus conjecture 2 is also widely believed to hold. Moreover, for any given dimension there is a precise conjectured value for the polynomial g: as the above calculation shows, it is essentially just the function $\widetilde{f}$ from conjecture 1, with appropriate values for the constants γ and A.

The behavior of our algorithm may now be stated more precisely as follows. Fix a dimension d and a polynomial g, and suppose first that conjecture 2 holds. Then, on inputs $\epsilon, \delta \in (0,1)$, the algorithm outputs a sequence of numbers $\widetilde{c}_1, \widetilde{c}_2, \widetilde{c}_3, \ldots$, such that, for each n, the time to output $\widetilde{c}_n$ is a polynomial function of n, ϵ^{-1} and $\log \delta^{-1}$ and, with probability at least $(1-\delta)$, $\widetilde{c}_n$ approximates c_n within ratio $(1+\epsilon)$. If, on the other hand,

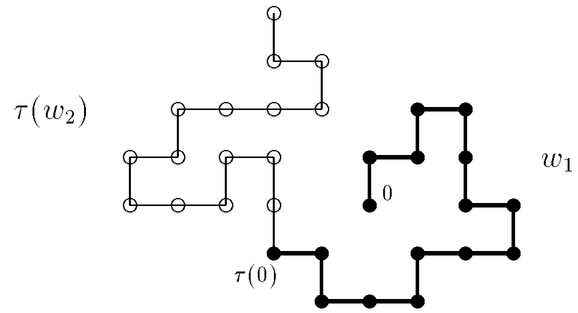


Figure 3.1: The concatenation of two self-avoiding walks.

the conjecture happens to fail for some value $n = n_0$, then with high probability an error will be reported and we will know that the algorithm is reliable in the region previously explored (i.e., for $n < n_0$), but may be unreliable for larger values of n. Moreover, in this case the algorithm will actually have discovered a counter-example to the conjecture for the polynomial g under consideration; since precise conjectured values for g exist, this in itself would be of substantial interest in the theory of self-avoiding walks. The details of the self-tester are described explicitly in section 3.3. Note that, in the presence of the self-tester, the answers output by our algorithm are always correct (with high probability), and the algorithm is guaranteed always to run in polynomial time.

The algorithm is in fact more flexible and can be used in addition to solve problem (ii) of section 3.1.1 by generating random self-avoiding walks of any specified length in the region where conjecture 2 holds: once the algorithm has output $\tilde{c}_n$, it provides a method for generating, in time polynomial in n and $\log \delta^{-1}$, a random self-avoiding walk of length n from a distribution whose variation distance from the uniform distribution is at most δ. This gives us a *fully-polynomial almost uniform generator* for self-avoiding walks (see definition 1.3.2), and can be used in the obvious fashion to obtain good statistical estimates in polynomial time of quantities such as the mean-square displacement.

In section 3.2 we present approximation algorithms which work assuming conjecture 2. In section 3.3 we show how to make the algorithms robust by adding a self-tester to verify the conjecture. In section 3.4 we present a more efficient version of the algorithm which was implemented to get the numerical estimates for $\tilde{c}_n$ tabulated in section 3.5.

3.2 The algorithms

Our goal is to design a fully-polynomial randomized approximation scheme and a fully-polynomial almost uniform generator for self-avoiding walks of length n. The following quantity associated with self-avoiding walks will play an important role in our algorithms.

For a fixed dimension d and each n, define

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

The quantity α_n is a lower bound on the probability that the concatenation of two walks w_1 and w_2 is self-avoiding, where w_1 and w_2 are chosen so that the sum of their lengths is at most n. Note that conjecture 2 says precisely that $\alpha_n \ge g(n)^{-1}$ for a polynomial g (which depends on d). We have the following:

Theorem 3.2.1 *For any fixed dimension d, there exists a randomized approximation scheme for self-avoiding walks that runs in time polynomial in $n, \epsilon^{-1}, \log \delta^{-1}$ and α_n^{-1}, and an almost uniform generator that runs in time polynomial in $n, \log \epsilon^{-1}$ and α_n^{-1}.*

It is interesting to observe that this result, combined with the asymptotic bound on c_n of Hammersley and Welsh [28] quoted in section 3.1.1, immediately gives us approximation algorithms for self-avoiding walks whose running time is sub-exponential. Specifically, the bound of [28] implies that $\alpha_n^{-1} = \mathrm{O}(a^{n^{1/2}})$ for some constant a, and closer inspection of the running time of the algorithms of theorem 3.2.1 (see section 3.2.3) reveals that the dependence on α_n^{-1} is linear. Thus we get a randomized approximation scheme and an almost uniform generator whose running times grow with n only as $\exp(\mathrm{O}(n^{1/2}))$.

If we assume conjecture 2, however, we get something much stronger. Since conjecture 2 asserts that $\alpha_n \ge g(n)^{-1}$ for a polynomial g, we immediately deduce the following.

Corollary 3.2.2 *Assuming conjecture 2, there exists a fully-polynomial randomized approximation scheme and a fully-polynomial almost uniform generator for self-avoiding walks in any fixed dimension d.* □

Our algorithms are based on randomly sampling walks of length n using Monte Carlo simulation of a series of successively larger Markov chains $M_1, ..., M_n$. In section 3.2.1 we define the nth Markov chain M_n, and in section 3.2.2 we derive a bound on its rate of convergence to stationarity. With this machinery in place, in section 3.2.3 we assemble the Markov chains into a single scheme that provides algorithms satisfying the requirements of theorem 1.

3.2.1 The Markov chain

As indicated in section 3.1.2, we consider a Markov chain that explores the space of self-avoiding walks by letting a walk expand and contract randomly over time, under the influence of a weighting parameter β. Rather than working with a single Markov chain and a global value of the parameter β, we incrementally construct Markov chains $M_1, M_2, \ldots$, the nth of which, M_n, has as its state space the set $\mathcal{X}_n = \bigcup_{i=0}^{n} \mathcal{S}_i$ of all self-avoiding walks of length at most n. The transition probabilities in M_n depend on parameters $\beta_1, ..., \beta_n \in (0, 1)$, discussed below.

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

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

$$P_n(w, w') = \begin{cases} \beta_{|w'|}/2d, & \text{if } w \prec_1 w'; \\ 1/2d, & \text{if } w' \prec_1 w; \\ r(w), & \text{if } w = w'; \\ 0, & \text{otherwise,} \end{cases} \tag{3.1}$$

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

Note that we may view M_n as a weighted random walk on the tree defined by the partial order $\prec$. This tree has the trivial walk of length 0 at the root, and the children of walk w are walks w' with $w \prec_1 w'$. Thus the tree has $n+1$ levels, the ith of which contains all walks of length $i-1$. The transition probability from any state to its parent is $1/2d$, and from a state at level i to each of its children is $\beta_{i+1}/2d$. In the case that $\beta_1 = \ldots = \beta_n \approx \mu^{-1}$ this is just the Markov chain used by Berretti and Sokal [3], but truncated at level n.

It is evident that the Markov chain M_n is irreducible (all states communicate) and aperiodic. This implies that it is ergodic, i.e., it converges asymptotically to a well-defined stationary distribution π_n over $\mathcal{X}_n$. It is straightforward to show the following:

Lemma 3.2.3 *The stationary distribution π_n of the Markov chain M_n is given by*

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

where Z_n is a normalizing factor.

Proof. It suffices to show that the chain is reversible with respect to the distribution π_n, i.e., that it satisfies the detailed balance condition (see equation (1.4)). This is readily verified from the definition of P_n given in (3.1). $\square$

Note that the stationary distribution is always uniform over all walks of a given length, for any choice of values of the parameters β_i. Moreover, by choosing the β_i carefully we can achieve a distribution over *lengths* which assigns sufficiently high weight to $\mathcal{S}_n$. Ideally, the value we want for β_i is the ratio c_{i-1}/c_i. (The fact that this ratio is always strictly less than 1 was proven surprisingly recently by O'Brien [52].) Of course, this is unrealistic since we do not know the quantities c_{i-1} and c_i—these are precisely what we are trying to compute—but we will see in section 3.2.3 how to determine good approximations to these ideal values before they are needed. For the moment, we consider the ideal behavior of the Markov chain assuming that each β_i is equal to c_{i-1}/c_i.

Under this assumption, lemma 3.2.3 says that the stationary probability of any walk $w \in \mathcal{X}_n$ is

$$\pi_n(w) = \frac{1}{Z_n} \prod_{i=1}^{|w|} \frac{c_{i-1}}{c_i} = \frac{1}{Z_n c_{|w|}}. \tag{3.2}$$

Thus the stationary distribution is uniform over lengths, and the probability of being at a walk of length i is $1/Z_n = 1/(n+1)$ for each i. It follows that the stationary distribution is reasonably well concentrated on $\mathcal{S}_n$, and uniform over $\mathcal{S}_n$. We may therefore, at least in principle, generate random self-avoiding walks of length n by simulating M_n until it has reached equilibrium, starting with, say, the empty walk, and outputting the final state if it has length n. We now need to show that the number of simulation steps required by the Markov chain is small. This is the key component of the running time of our algorithms and is quantified in the next subsection.

3.2.2 The mixing time

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

In recent years several useful analytical tools have been devised for analyzing the mixing time of complex Markov chains of this kind. We make use of the idea of "canonical paths", first developed in [34, 59]. Consider an ergodic, reversible Markov chain with state space X, transition probabilities P and stationary distribution π. We can view the chain as a weighted undirected graph G with vertex set X and an edge between each pair of vertices (states) x, y for which $P(x,y) > 0$. We give each oriented edge $e = (x,y)$ a "capacity" $Q(e) = Q(x,y) = \pi(x)P(x,y)$; note that, by detailed balance, $Q(x,y) = Q(y,x)$.

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

$$\rho(\Gamma) = \max_e \frac{1}{Q(e)} \sum_{\gamma_{xy} \ni e} \pi(x)\pi(y) \tag{3.3}$$

where the maximization is over oriented edges e. Thus ρ measures the maximum loading of any edge e by paths in Γ as a fraction of its capacity $Q(e)$, where the path from x to y carries "flow" $\pi(x)\pi(y)$. Note that the existence of a collection of paths Γ for which $\rho(\Gamma)$ is small implies an absence of bottlenecks in the graph, and hence suggests that the Markov chain should be rapidly mixing. This intuition can be formalized and a bound obtained on the mixing time in terms of the quantity $\rho = \min_\Gamma \rho(\Gamma)$, using a measure known as *conductance* [60]. However, we can get a slightly sharper bound in this case by following an idea of Diaconis and Stroock [12] and using the alternative measure $\overline{\rho} = \min_\Gamma \rho(\Gamma)\ell(\Gamma)$, where $\ell(\Gamma)$ is the maximum length of a path in Γ. The appropriate version of this bound can be found by combining Proposition 1 and Corollary 6 of [58] and is stated precisely in the theorem below.

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

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

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

Theorem 3.2.4 [58] *For an ergodic, reversible Markov chain with stationary distribution π, we have*

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

(where all logarithms are assumed to be base e). □

We now use theorem 3.2.4 to show that the mixing time of the Markov chain M_n can be bounded in terms of the quantity α_n defined at the beginning of section 3.2. Assuming conjecture 2, this will imply that the Markov chain is rapidly mixing. For simplicity we will work with the idealized version of M_n discussed at the end of section 3.2.1, in which each β_i is exactly equal to c_{i-1}/c_i. It should be clear that our analysis is not unduly sensitive to small perturbations in the values of the β_i.

Theorem 3.2.5 *For the Markov chain M_n, starting at the empty walk $\mathbf{0}$, we have*

$$\tau_{\mathbf{0}}(\epsilon) \le 4dn^2\alpha_n^{-1}\Big(\log n + \log \epsilon^{-1}\Big).$$

Proof. From (3.2) we have that $\pi_n(\mathbf{0}) = 1/(n+1)$. Also, since the graph corresponding to the Markov chain M_n is a tree, there is only one choice of (simple) paths between each pair of vertices; we will denote this collection of paths $\Gamma = \{\gamma_{xy}\}$. Since the depth of the tree is n, we have $\ell(\Gamma) = 2n$. Therefore, the result will follow from theorem 3.2.4 if we can show that $\rho(\Gamma) \le K'dn\alpha_n^{-1}$ for some constant K'.

Now let e be any edge of the tree, and suppose the endpoints of e are a walk w of length k and a walk w' of length $k+1$. Let S be the subtree rooted at w', and $\overline{S} = \mathcal{X}_n - S$. Since e is a cut edge, it is clear that (3.3) becomes

$$\rho(\Gamma) = \max_e \; Q(e)^{-1}\pi_n(S)\pi_n(\overline{S}). \tag{3.4}$$

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

Now we have

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

and

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

where the first inequality follows from the sub-Cayley property of the tree. Putting this together, we see that $Q(e)^{-1}\pi_n(S)\pi_n(\overline{S}) \le Q(e)^{-1}\pi_n(S) \le 2dn\alpha_n^{-1}$. Since e was arbitrary, (3.4) now gives us the required upper bound on $\rho(\Gamma)$. □

Remark. A similar bound on the mixing time of the Berretti-Sokal Markov chain was obtained using ad-hoc methods by Sokal and Thomas [62]. Again the essential feature that makes the argument work is the sub-Cayley property of the tree underlying the chain. A rather weaker bound was obtained by Lawler and Sokal [46], using the discrete Cheeger inequality or conductance. This latter proof is very similar in spirit to the one above; essentially, the effect is to replace $\overline{\rho}$ by ρ^2 in the bound of theorem 3.2.4. □

3.2.3 The overall algorithm

In this subsection, we show how to assemble the family of Markov chains just described into a single algorithm that outputs a sequence of numbers $\{\widetilde{c}_n\}$, each of which is a good estimate of the corresponding c_n. The accuracy of the estimates is controlled by two parameters, ϵ and δ, exactly as in the definition of a randomized approximation scheme (definition 1.3.1). We shall see that the algorithm provides both an approximation scheme and an almost uniform generator with the properties claimed in theorem 3.2.1.

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

The algorithm, spelled out in detail in figure 3.2, works in a sequence of stages corresponding to the iterations of the **for**-loop. We call stage n of the algorithm *good* if it runs to completion and computes a value β_n that approximates the value c_{n-1}/c_n within ratio $(1+\epsilon/4n^2)$, where ϵ, δ are the accuracy and confidence inputs.

Let us consider the operation of stage n in detail. To compute a good approximation β_n of the ratio c_{n-1}/c_n, we randomly sample walks of length $n-1$ using the Markov

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

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

> using M_{n-1}, generate a sample of size $2nT_n$ from (close to) the distribution π_{n-1} over $\mathcal{X}_{n-1}$
>
> let Y be the set of walks in the sample with length $n-1$
>
> $\mathrm{ext}(Y) = \sum_{w \in Y} |\{w' \in \mathcal{S}_n : w \prec_1 w'\}|$
>
> **if** $|Y| < T_n$ or $\mathrm{ext}(Y) = 0$ **then abort**
>
> **else** set $\beta_n = |Y|/\mathrm{ext}(Y)$
>
> **output** $\tilde{c}_n = \tilde{c}_{n-1}/\beta_n$

Figure 3.2: The algorithm

chain M_{n-1} and estimate the average number of one-step extensions of a walk: we can compute the number of one-step extensions of a given walk by explicitly checking each of the $2d-1$ possibilities. Note that, for a random walk, this is a bounded random variable taking values in $[0, 2d-1]$ with mean at least 1 (since $c_n > c_{n-1}$). The sample size is controlled by the parameter T_n. A simple generalization of the 0/1-estimator theorem (see [39]) to handle non-negative, bounded random variables shows that T_n need not be too large to obtain a good estimate with sufficiently high probability. Finally, since we are in fact sampling from the larger set $\mathcal{X}_{n-1}$, we need to generate a sample of size $2nT_n$ to ensure that, with high probability, we get at least T_n walks of length $n-1$; that this sample is large enough follows from the fact that, by (3.2), in the stationary distribution of the chain M_{n-1} walks of length $n-1$ have weight $1/n$.* In the algorithm of figure 3.2, we abort in the unlikely event that the sample does not contain enough walks of length $n-1$.

Summarizing the above discussion, we get:

Lemma 3.2.6 *In the algorithm of figure 3.2, assuming that stages* $1, 2, \ldots, n-1$ *are good, stage* n *is good with probability at least* $(1 - \delta/2n^2)$*, provided the sample size* T_n *is at least* $cn^4\epsilon^{-2}(\log n + \log \delta^{-1})$ *for a suitable constant* c *(which depends on the dimension* d*).* □

The algorithm is designed so that, assuming all previous stages $1, 2, \ldots, n-1$ are good, stage n will be good with probability at least $(1 - \delta/2n^2)$. The reason for this requirement

*Actually the Markov chain we are simulating here is not precisely that analyzed in sections 3.2.1 and 3.2.2, since the parameters β_i will differ slightly from their ideal values. However, it should be clear from lemma 3.2.3 that the stationary distribution is always uniform within each level of the tree, and that, if all previous stages are good, then the distribution over levels of the tree differs from the uniform distribution by at most a constant factor.

is the following. If all stages $1, 2, \ldots, n$ are good, then the value $\tilde{c}_n = \prod_{i=1}^n \beta_i^{-1}$ output by the algorithm at the end of stage n approximates $\prod_{i=1}^n (c_i/c_{i-1}) = c_n$ within ratio $\prod_{i=1}^n (1 + \epsilon/4i^2) \leq 1 + \epsilon$; moreover, this happens with probability at least $\prod_{i=1}^n (1 - \delta/2i^2) \geq 1 - \delta$. Thus we get a randomized approximation scheme for c_n, which was one of our principal goals. Moreover, by the end of stage n we have computed values β_i for $1 \leq i \leq n$; thus we have constructed a Markov chain M_n which we can simulate to generate random self-avoiding walks of any length up to n. This was our second principal goal.

The running time of stage n of the algorithm is dominated by $2nT_n = 2cn^5\epsilon^{-2}(\log n + \log \delta^{-1})$ times the time required to produce a single sample from M_{n-1}. From our analysis in the previous subsection (theorem 3.2.5), this latter time is $\mathrm{O}\left(n^2 \alpha_n^{-1} (\log n + \log \epsilon^{-1})\right)$ for any fixed dimension d.† Run to the nth stage, the algorithm is therefore an approximation scheme satisfying the requirements of theorem 3.2.1. (Note that the exponent of n in the running time could be improved by a more refined statistical analysis.) By the same reasoning, simulating the Markov chain M_n for $\mathrm{O}\left(n^2 \alpha_n^{-1} (\log n + \log \epsilon^{-1})\right)$ steps gives us the almost uniform generator claimed in theorem 3.2.1.

3.3 Making the algorithm self-testing

In this section we show how to place the algorithm of the previous section on a firmer theoretical basis by replacing our assumption of conjecture 2 by an algorithmic test of the conjecture. This is a particular instance of what we believe is a generally useful idea of using *self-testing* to make an algorithm whose correctness depends on a conjecture more robust.

Recall that we have reduced the problem of constructing polynomial time approximation algorithms for self-avoiding walks to that of verifying a single widely believed conjecture about the behavior of the walks. An important feature of this reduction is the structure of the conjecture. Conjecture 2 bounds the probability that one can glue together two random self-avoiding walks to produce a new self-avoiding walk; since the algorithm also lets us generate random self-avoiding walks, we can actually estimate this probability. For any specified polynomial g, this allows us to verify the conjecture (and therefore also the algorithm itself) for a new value of n by using the algorithm in the region in which it has already been tested. This is precisely the idea behind the self-tester which we introduce in this section.

We showed in the last section how to construct a sequence of Markov chains for uniformly generating and counting self-avoiding walks of successively longer lengths. The running time of these algorithms is polynomial in the walk length n and the unknown parameter α_n^{-1}; this quantity enters into the time bound because it governs the mixing time of the Markov chains (see theorem 3.2.5). We then appealed to conjecture 2 to argue that α_n^{-1} is itself polynomially bounded in n. The idea behind the self-tester is to obtain a good estimate for α_n^{-1} in advance, so that we know how long to simulate our Markov chains to ensure our samples are sufficiently close to the stationary distribution. This will give

† Once again, we should point out that the analysis of theorem 3.2.5 refers to the idealized Markov chain in which all values β_i are exact. However, it is a simple matter to check that, assuming all stages are good, the effect on the mixing time of these small perturbations of the β_i is at most a constant factor.

$\tilde{\alpha}_n = \tilde{\alpha}_{n-1}$

for $i = 1, 2, \ldots, n-1$ **do**

  repeat t **times**

    generate $u_i \in \mathcal{S}_i$ u.a.r.

    generate $v_i \in \mathcal{S}_{n-i}$ u.a.r.

$$X_i = \begin{cases} 1 & \text{if } u_i \circ v_i \in \mathcal{S}_n\,; \\ 0 & \text{otherwise} \end{cases}$$

  $q_{n,i} = \sum_i X_i / t$

  $\tilde{\alpha}_n = \min\{\tilde{\alpha}_n, q_{n,i}/2\}$

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

then output "Warning: conjecture fails"

else continue

Figure 3.3: The self-tester

us a probabilistic *guarantee* that the algorithm is correct, independent of conjecture 2. Moreover, by examining the growth rate of successive values α_n^{-1}, we can simultaneously *test* conjecture 2. The algorithm will proceed successfully for as long as we never exceed some projected upper bound $g(n)$ on α_n^{-1}; however, should α_n^{-1} grow too quickly we will detect this fact and we will have found a counter-example to the conjecture.

More precisely, after each stage n of the algorithm of the previous section, we use the procedure shown in figure 3.3 to compute a quantity $\tilde{\alpha}_n$ such that, with high probability, $\alpha_n/4 \le \tilde{\alpha}_n \le \alpha_n$. The conservative estimate $\tilde{\alpha}_n^{-1}$ is then used in place of α_n^{-1} in the next stage when computing the mixing time of the Markov chain M_n from theorem 3.2.5. We also test conjecture 2 by comparing our estimates $\tilde{\alpha}_n^{-1}$ at each stage with (a constant multiple of) the projected polynomial upper bound $g(n)$. The algorithm with the self-tester has the following properties with high probability:

(i) if $\alpha_n^{-1} \le g(n)$ (i.e., conjecture 2 holds), then the algorithm outputs a reliable numerical answer;

(ii) if $\alpha_n^{-1} > 4g(n)$ (i.e., conjecture 2 fails "badly"), then the algorithm outputs an error message;

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

Furthermore, in every case the algorithm runs in polynomial time.

Our method for computing the estimate $\widetilde{\alpha}_n$ again uses random sampling from the stationary distribution of the Markov chain M_{n-1}. The idea is to generate self-avoiding walks of lengths i and $n-i$ uniformly at random, and thus estimate the probability that such a pair can be glued together to form a walk of length n that is self-avoiding. The details are given in figure 3.3. Elementary statistics show that the sample size t required to ensure that the estimate $\widetilde{\alpha}_n$ is within the bounds specified above with high probability is a polynomial function of n, so the self-testing portion of the algorithm also runs in polynomial time.

Theorem 3.3.1 *The algorithm of section 3.2 with the self-tester incorporated runs in time polynomial in n, ϵ^{-1} and $\log \delta^{-1}$, and satisfies properties (i)-(iii) above.* $\square$

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

The idea of a tester has been used before, but in a much more restrictive sense. For example, Berretti and Sokal [3] propose testing possible "errors in scaling" due to the conjecture that $f(n) \approx An^{\gamma-1}$ by trying other specific polynomial forms for $f(n)$. This gives evidence that $f(n)$ might be of the correct form, but falls short of proving it probabilistically. In contrast, the tester we present is designed to verify exactly the conjecture we require, and therefore offers precisely quantified statistical evidence that our algorithm is operating as we expect.

3.4 Improved time bounds

In sections 3.2 and 3.3 we presented a testable algorithm whose running time is polynomially bounded. In this section we are concerned with improving the efficiency. The numerical results which are presented in the next section are based on this improved algorithm. A careful honing of the analysis of the algorithm presented in figure 3.2 already produces significant improvements in the running time. We restrict our discussion to the conceptual changes to the algorithm which yield even greater improvements in the running time.

As before we will run the algorithm in stages, where the nth stage is based on simulating a Markov chain whose state space is the set of self-avoiding walks of length at most $n-1$. The inputs to the nth stage are parameters $\beta_1, ..., \beta_{n-1}$, and again we will produce a new numerical estimate of the number of self-avoiding walks of length n and a new parameter β_n. The modification to the algorithm involves how we generate these outputs. Recall that previously we calculated our estimate β_n and then set $\tilde{c}_n = \tilde{c}_{n-1}/\beta_n$. In contrast, we now calculate $\tilde{c}_n$ directly and then set β_n to be the quotient $\tilde{c}_n/\tilde{c}_{n-1}$. To calculate $\tilde{c}_n$, we will take the median of $\hat{k}_n$ independent random variables, $Z_1, ..., Z_{\hat{k}_n}$; each Z_k is an estimate for $\tilde{c}_n$. To determine Z_k, we again take the product of n independent random variables $Z_{k,1}, ..., Z_{k,n}$, where each $Z_{k,i}$ is an estimate of the number of 1-step extensions of a walk of length i. In stage n we update the values of all of these random variables.

The improvement in the running time comes from two observations. The first is a standard trick for reducing the number of samples required to get a good estimate of the product of random variables (see, e.g., [13]). Rather than requiring that the variance of all n random variables be *extremely* small (so that the product will be close to its mean), we bound the variance of the product directly. This allows us to reduce the number of samples needed since each factor no longer needs to be quite as good; the deviations tend to cancel each other out.

The second observation is more specialized to features of our process. Previously at stage n we used only sample walks of length n; all others were rejected. In our new implementation, we use each sample walk of length $i < n$ to update our estimate for $Z_{k,i}$. By the nth stage we have taken many samples to calculate $Z_{k,i}$ when $i \ll n$, and so $Z_{k,i}$ will be very accurate. Consequently, we are able to tolerate a relatively weaker estimate of the $Z_{k,j}$ where j is close to n; the product will still be a good estimate of c_n. In successive stages our estimates for all of the $Z_{k,i}$ improve.

We find that instead of requiring a polynomial number of samples of size n, using these two ideas we only require $O(\log n)$ samples of each size in the nth stage. Since the Markov chains are designed so that the outputs are uniformly distributed over levels, this will lead to a large improvement in the running time. The new algorithm is presented in figure 3.4.

The use of the median in the algorithm is a standard trick for boosting the confidence parameter. We will show that for each of the $\hat{k}$ independent trials, the estimate Z_k will approximate c_n to within a factor of $1+\epsilon$ with probability at least 3/4. From this it follows that the median of $\{Z_1, ..., Z_{\hat{k}}\}$ will approximate c_n to within $1+\epsilon$ with probability at least $1 - \delta/(n \log^2 n)$ if $\hat{k}$ is chosen to be at least $\hat{c} \log(n \log^2 n/\delta)$, for some constant $\hat{c}$. Therefore the probability that *all* the outputs are correct is at least $1 - \sum_{i=1}^{\infty} \delta/(i \log^2 i) \geq 1 - \delta$.

The "buffer data" alluded to in the algorithm is used to get around the anomaly arising from the fact that as n increases, so will the number of trials required to compute the median. By taking an additional sample of size $4T_n$ at each stage of the algorithm, we can remedy this problem. Whenever $\hat{t}$ increases from the previous stage, we use m_i buffer samples of size i (for each i) to evaluate $\mathrm{est}(i, n-1)$ and $\mathrm{samp}(i, n-1)$ for the $\hat{k}$th trial; we then proceed as usual. With high probability we will have a sufficient number of buffer samples of each size.

We are now in a position to analyze the algorithm. Call a stage n "good" if it runs to completion and computes an estimate that approximates c_n to within a factor of $1+\epsilon$. Clearly, if all of the stages $1, ..., n-1$ are good, then the stationary distribution of the nth Markov chain will be close to π, where $\pi(w) = ((n+1)c_{|w|})^{-1}$. As before, our analysis will not be unduly affected by the small deviations in these probabilities.

To analyze the behavior of stage n, we first assume that all previous stages are good and we show that stage n is also good with high probability. Let $\mathrm{E}[\cdot]$ be the expected value of a random variable, and let $\mathrm{Var}[\cdot]$ be its variance. Note that by stage n the number of samples of size $i \leq n$ is at least

$$m_{i,n} = \sum_{j=i}^{n} c\epsilon^{-2} \lceil \log j \rceil \geq c\epsilon^{-2}(n-i+1)\lceil \log(n/2) \rceil. \tag{3.5}$$

We show that this is sufficient to bound the variance of Z_k for an appropriate choice of c. The following two lemmas perform the role of lemma 3.2.6 for the original algorithm.

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

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

  $m_n = c\epsilon^{-2}\lceil \log n \rceil; \quad T_n = 2cnm_n; \quad \hat{k}_n = \lceil \hat{c} \log(n \log^2 n/\delta) \rceil$

  if $\hat{k}_n > \hat{k}_{n-1}$ use buffer data to create a new sample

    (**if** the buffer sample is too small **then abort**)

  for $1 \leq k \leq \hat{k}_n$ **do**

    using M_{n-1}, generate a sample of size T_n from (close to) the distribution π_{n-1} over $\mathcal{X}_{n-1}$

    for $1 \leq i \leq n$ **do**

      let Y_i be the set of walks in the sample with length $i - 1$

      (**if** $|Y_i| < m_n$ **then abort**)

      $\text{ext}(i, n) = \text{ext}(i, n-1) + \sum_{w \in Y_i} |\{w' \in \mathcal{S}_n : w \prec_1 w'\}|$

      $\text{samp}(i, n) = \text{samp}(i, n-1) + |Y_i|$

      $Z_{k,i} = \text{ext}(i, n)/\text{samp}(i, n)$

    $Z_k = \prod_i Z_{k,i}$

  output $\tilde{c}_n = \text{MEDIAN}(Z_1, Z_2, \ldots, Z_{\hat{k}_n})$

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

  using M_{n-1}, generate buffer sample of size $4T_n$

Figure 3.4: The improved algorithm

Lemma 3.4.1 *Assume that in the algorithm of figure 3.4 stages* $1, 2, \ldots, n-1$ *are good. Then in stage* n*, for each* k*,*

$$\frac{\text{Var}[Z_k]}{\text{E}[Z_k]^2} \leq \epsilon^2/4,$$

assuming stage n *runs to completion.*

Proof. The random variable Z_k is the product of n independent random variables $Z_{k,1} \times Z_{k,2} \times \ldots \times Z_{k,n}$, so $\text{E}[Z_k] = \prod_{i=1}^{n} \text{E}[Z_{k,i}]$. Using the fact that $Z_{k,i}$ is the sum of $m_{i,n}$ independent random variables between 0 and $2d$ we can bound the variance.

$$\frac{\text{Var}[Z_k]}{\text{E}[Z_k]^2} = \prod_{i=1}^{n} \Big(\frac{\text{E}[Z_{k,i}^2]}{\text{E}[Z_{k,i}]^2} \Big) - 1$$

$$
\begin{aligned}
&= \prod_{i=1}^{n}\Big(\frac{\mathrm{Var}[Z_{k,i}]}{\mathrm{E}[Z_{k,i}]^2}+1\Big)-1 \\
&\le \prod_{i=1}^{n}\Big(\frac{2d}{m_{i,n}\mathrm{E}[Z_{k,i}]}+1\Big)-1 \\
&\le \prod_{i=1}^{n}\Big(\frac{2dp}{m_{i,n}}+1\Big)-1, \qquad (3.6)
\end{aligned}
$$

if $p^{-1} \le \min_i \mathrm{E}[Z_{k,i}]$. Now, to get a bound on the right hand side of this inequality, we take the following natural logarithms

$$
\begin{aligned}
\log\Big(\prod_{i=1}^{n}\Big(\frac{2dp}{m_{i,n}}+1\Big)\Big) &= \sum_{i=1}^{n}\log\Big(\frac{2dp}{m_{i,n}}+1\Big) \\
&\le \sum_{i=1}^{n}\frac{2dp}{m_{i,n}} \\
&\le \frac{2dp}{c}\epsilon^2\sum_{i=1}^{n}\Big(\frac{1}{n-i+1}\Big)\Big(\frac{1}{\log n/2}\Big) \\
&\le c'\epsilon^2 \\
&\le \log(c'(\mathrm{e}+1)\epsilon^2+1), \qquad (3.7)
\end{aligned}
$$

where we use the fact that for small $x > 0, x/(\mathrm{e}+1) \le \log(x+1) \le x$. The second inequality follows from equation (3.5) for the number of samples of each size. From the monotonicity of the log function, equation (3.7) gives us that

$$
\prod_{i=1}^{n}\Big(\frac{2dp}{m_{i,n}}+1\Big) \le c'(\mathrm{e}+1)\epsilon^2+1. \qquad (3.8)
$$

We can choose c so that $c'/(\mathrm{e}+1) \le 1/4$. Substituting equation (3.8) back into equation (3.6), we find

$$
\frac{\mathrm{Var}[Z_k]}{\mathrm{E}[Z_k]^2} \le \epsilon^2/4.
$$

□

This bound on the variance allows us to analyze the performance of this algorithm using Chebyshev's inequality in the next proof.

Lemma 3.4.2 *In the algorithm of figure 3.4, assuming that stages 1,2,...,$n-1$ are good, stage n is good with probability at least* $1-\delta/(2n\log^2 n)$.

Proof. Assuming that all the previous stages are good, the outputs from Markov chains M_{n-1} are reliable. We will bound the probability that stage n is not good, i.e., our estimate for c_n is not within a factor of $1+\epsilon$. Chebyshev's inequality tells us that for each independent trial k,

$$
\Pr[|Z_k-\mathrm{E}[Z_k]| > \mathrm{E}[Z_k](1+\epsilon)] \le \frac{\mathrm{Var}[Z_k]}{\mathrm{E}[Z_k]^2\epsilon^2} \le 1/4
$$

from lemma 3.4.1 above. Using a standard technique for boosting the confidence parameter, since we let $\tilde{c}_n$ be the median of $\hat{k}_n = c' \log(n \log^2 n/\delta)$ independent trials, we get

$$\Pr[|\tilde{c}_n - c_n| > c_n(1+\epsilon)] \leq \delta/(n \log^2 n).$$

□

Using the algorithm presented in figure 3.4, only $O(n\epsilon^{-2} \log n(\log n + \log \delta^{-1}))$ samples are required in each stage. This is a large improvement over the corresponding bound from lemma 3.2.6 which included a factor of n^4. Summarizing our discussion, we have the following.

Theorem 3.4.3 *The algorithm in figure 3.4 outputs estimates $\tilde{c}_1, \tilde{c}_2, \ldots$ such that all the estimates $\tilde{c}_n$ are within $1+\epsilon)$ of the number of self-avoiding walks of length n with probability at least $1-\delta$. The running time for each incremental value of n is $O(n^3 \alpha_n^{-1} \epsilon^{-2} \log n \log \epsilon^{-1}(\log n + \log \delta^{-1}))$.*

3.5 Numerical results

In this section we present our estimates for the number of self-avoiding walks on the 2- and 3-dimensional cartesian lattices. These were obtained by implementing the algorithm presented in section 3.4 with improvements in the constants.

There has already been substantial effort in the physics community to count the number of self-avoiding walks *exactly*, especially in two dimensions. Using various ingenious (although exponential-time) algorithms the number of self-avoiding walks on the 2-dimensional lattice is now known to length 50 [10]. We include the published results up to length 39 for comparison with our estimates [48, 10]. In three dimensions, where there are far more self-avoiding walks than in two dimensions, exact enumerations have been less successful; the exact number of walks in the 3-dimensional cartesian lattice have only been computed to length 21 [48].

In contrast, notice that our algorithm actually performs better in higher dimensions (assuming conjecture 1 which suggests that the factor α^{-1} is expected to decrease as the dimension increases). Consequently, we have concentrated our efforts on the 3-dimensional lattice which has the greatest physical significance.

The program was implemented in C for the 2- and 3-dimensional lattices. The accuracy parameter ϵ was chosen to be .05 and the confidence parameter δ was chosen to be .01 (i.e., we expect all of our answers to be within a factor of 1 ± 0.05 of the real counts with 99% reliability). The 2-dimensional problem ran for just under two weeks on a 4-processor SGI Challenge. The 3-dimensional problem ran for a week and a half on an 8-processor SGI Challenge. The results of the computation are presented in figures 3.5 and 3.5. Comparing with the exact numbers, where they exist, we see that the program actually achieves a much greater accuracy than the 5% relative error guaranteed by the analysis, typically less than 0.6 %. (This error was calculated relative to the larger number, i.e. $100|c_n - \tilde{c}_n|/\max(c_n, \tilde{c}_n)$.

Walk Length	Exact Count	Estimates	Relative Error (%)
1	4	4	0
2	12	12	0
3	36	36	0
4	100	99.9864	0.0136
5	284	283.976	0.0084507
6	780	780.67	0.0858237
7	2172	2173.59	0.0731509
8	5916	5919.81	0.0643602
9	16268	16266.1	0.0116794
10	44100	44161.7	0.139714
11	120292	120473	0.150241
12	324932	325934	0.307424
13	881500	885486	0.450148
14	2.37444e+06	2.38211e+06	0.321816
15	6.4166e+06	6.4414e+06	0.385072
16	1.72453e+07	1.73229e+07	0.447777
17	4.64667e+07	4.65599e+07	0.200224
18	1.24659e+08	1.25075e+08	0.332815
19	3.35117e+08	3.3596e+08	0.251036
20	8.97697e+08	8.9852e+08	0.0915768
21	2.40881e+09	2.41378e+09	0.206066
22	6.44456e+09	6.45119e+09	0.102764
23	1.72666e+10	1.72786e+10	0.0693701
24	4.61464e+10	4.61653e+10	0.0409457
25	1.23481e+11	1.23373e+11	0.08775
26	3.29712e+11	3.29722e+11	0.00294605
27	8.81317e+11	8.7936e+11	0.22211
28	2.35138e+12	2.34174e+12	0.409912
29	6.2794e+12	6.24908e+12	0.482789
30	1.6742e+13	1.66857e+13	0.33603
31	4.46738e+13	4.45826e+13	0.204184
32	1.19035e+14	1.18761e+14	0.230183
33	3.17407e+14	3.16251e+14	0.364075
34	8.45279e+14	8.42589e+14	0.318247
35	2.25253e+15	2.23976e+15	0.567098

Figure 3.5: The number of self-avoiding walks in 2 dimensions

Walk Length	Exact Count	Estimates	Relative Error (%)
36	5.99574e+15	5.95715e+15	0.643632
37	1.59689e+16	1.58591e+16	0.68729
38	4.24868e+16	4.22069e+16	0.658678
39	1.13102e+17	1.12524e+17	0.510759
40	-	2.98747e+17	-
41	-	7.96097e+17	-
42	-	2.12048e+18	-
43	-	5.64604e+18	-
44	-	1.506e+19	-
45	-	4.00693e+19	-
46	-	1.06809e+20	-
47	-	2.83977e+20	-
48	-	7.59756e+20	-

Figure 3.6: The number of self-avoiding walks in 2 dimensions (cont.)

Walk Length	Exact Count	Estimates	Relative Error (%)
1	6	6	0
2	30	30	0
3	150	150	0
4	726	726.092	0.0126706
5	3534	3533.47	0.0149972
6	16926	16930.4	0.0259888
7	81390	81392.2	0.00270296
8	387966	388358	0.100938
9	1.85389e+06	1.85468e+06	0.0428106
10	8.80988e+06	8.81417e+06	0.0486943
11	4.19342e+07	4.20031e+07	0.164155
12	1.98843e+08	1.99187e+08	0.172832
13	9.43974e+08	9.4585e+08	0.198324
14	4.46891e+09	4.47983e+09	0.243722
15	2.11751e+10	2.12181e+10	0.20244
16	1.00122e+11	1.00373e+11	0.250191
17	4.7373e+11	4.74175e+11	0.093794
18	2.23772e+12	2.24042e+12	0.120349
19	1.0576e+13	1.05863e+13	0.0969818
20	4.99173e+13	4.99634e+13	0.0922118
21	2.3571e+14	2.3585e+14	0.0593214
22	-	1.11191e+15	-
23	-	5.24224e+15	-
24	-	2.47294e+16	-
25	-	1.16544e+17	-
26	-	5.49127e+17	-
27	-	2.58801e+18	-
28	-	1.21876e+19	-
29	-	5.74323e+19	-
30	-	2.70513e+20	-
31	-	1.27351e+21	-
32	-	5.98999e+21	-
33	-	2.82177e+22	-
34	-	1.32874e+23	-
35	-	6.25858e+23	-

Figure 3.7: The number of self-avoiding walks in 3 dimensions

Walk Length	Exact Count	Estimates	Relative Error (%)
36	-	2.94424e+24	-
37	-	1.38338e+25	-
38	-	6.50602e+25	-
39	-	3.06604e+26	-
40	-	1.44248e+27	-
41	-	6.78155e+27	-
42	-	3.18745e+28	-
43	-	1.49918e+29	-
44	-	7.05367e+29	-
45	-	3.30457e+30	-
46	-	1.55367e+31	-
47	-	7.31748e+31	-
48	-	3.44242e+32	-
49	-	1.61518e+33	-
50	-	7.59255e+33	-
51	-	3.558e+34	-
52	-	1.66594e+35	-
53	-	7.81908e+35	-
54	-	3.68407e+36	-
55	-	1.73431e+37	-

Figure 3.8: The number of self-avoiding walks in 3 dimensions (cont.)

3.6 Concluding remarks and open problems

The most obvious and compelling open problem arising from the work in this chapter is verifying conjecture 2. This would constitute a substantial breakthrough in the classical theory of self-avoiding walks. However, it is less well studied than conjecture 1, and its more elementary combinatorial nature should make this task more feasible. Our results show that proving conjecture 2 for *any* polynomial g would yield the first provably polynomial-time Monte Carlo approximation algorithms for self-avoiding walks.

In addition, we predict that there are other natural applications for using self-testing to convert heuristics into robust algorithms. For example, a likely candidate from statistical mechanics is uniformly generating and counting *lattice trees*. A lattice tree is an acyclic lattice animal which is made up of $n-1$ lattice edges. Lattice trees model branched polymers in dilute solution and are believed to fall in the same universality class as lattice animals [67]. There is strong evidence that the expression for the number of lattice trees of size n has a similar form to conjecture 1. It is likely that the techniques used here to design a testable algorithm for self-avoiding walks can be extended to this wider class of lattice animals.

Another feature discussed in this chapter which is likely to lead to approximation algorithms for other lattice structures is the use of sub-Cayley trees for Monte Carlo simulations.

Again taking lattice trees as an example, imagine describing each tree by the unique path created by a breadth-first search traversal of its edges. There is a natural sub-Cayley tree which describes the set of these paths during all stages of the breadth-first search. By adding appropriate weights, we can modify the sub-Cayley tree so as to guarantee that at stationarity those paths which represent a complete traversal of a lattice tree will have significant weight. Yet, as usual, it is difficult to guarantee that we reach stationarity rapidly. The sub-Cayley feature of the graph underlying this or some closely related Markov chain may be the a useful tool for demonstrating rapid convergence.

Finally, notice that we have restricted our attention throughout this chapter to the unweighted problem in which we want to sample each walk of a given length with equal probability. A generalization worth exploring is sampling self-avoiding walks with attractive or repulsive forces among the bonds. We associate with each self-avoiding walk a weight describing how "spread out" or "tightly wound" the walk is, and we want to sample a walk proportional to its weight. In addition to the more obvious applications to long polymer chains, this problems arises in computational biology in the context of *protein prediction*. The sites of the self-avoiding walk represent carbon atoms of a long protein chain, and forces between carbon atoms which are adjacent in the lattice, but not along the protein, determine the structure of the protein. Biologists are interested in generating such "potential" protein structures according to their likelihood in order to infer properties about actual proteins which are difficult to probe directly[47].

Bibliography

[1] Aldous, D. Some inequalities for reversible Markov chains. *Journal of the London Mathematical Society (2)* **25** (1982), pp. 564–576.

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

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

[4] Bhattacharjee, S.M., Nagle, J.F., Huse, D.A. and Fisher, M.E. Critical behaviour of a three-dimensional dimer model. *Journal of Statistical Physics* **32** (1983), pp. 361–374.

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

[6] Bondy, J.A. and Welsh, D.J.A. A note on the monomer dimer problem. *Proceedings of the Cambridge Philosophical Society* **62** (1966), pp. 503–505.

[7] Broder, A.Z. How hard is it to marry at random? (On the approximation of the permanent). *Proceedings of the 18th ACM Symposium on Theory of Computing* (1986), pp. 50–58. Erratum in *Proceedings of the 20th ACM Symposium on Theory of Computing* (1988), pp. 551.

[8] Cipra, B.A. An introduction to the Ising model. *American Mathematical Monthly* **94** (1987) pp. 937–959.

[9] Cohen, E.G.D., de Boer, J. and Salsburg, Z.W. A cell-cluster theory for the liquid state II. *Physica* **XXI** (1955), pp. 137–147.

[10] Conway, A.R., Enting, I.G., Guttmann, A.J. Algebraic techniques for enumerating self-avoiding walks on the square lattice. *Journal of Physics A* **26** (1993) pp. 1519-1552.

[11] Diaconis, P. *Group representations in probability and statistics.* Lecture Notes Monograph Series Vol. 11, Institute of Mathematical Statistics, Hayward, California, 1988.

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

[13] Dyer, M. and Frieze, A. Computing the volume of convex bodies: a case where randomness provably helps. *Probabilistic Combinatorics and its Applications,* Proceedings of AMS Symposia in Applied Mathematics Volume 44 (1994) pp. 123-170.

[14] Elser V. Solution of the dimer problem on a hexagonal lattice with boundary. *Journal of Physics A: Math. Gen.* **17** (1984), pp. 1509–1513.

[15] Fisher, M.E. Statistical mechanics of dimers on a plane lattice. *Physics Review* **124** (1961), pp. 1664–1672.

[16] Fisher, M.E. and Stephenson, J. Statistical mechanics of dimers on a plane lattice II. Dimer correlations and monomers. *Physics Review* **132** (1963), pp. 1411-1431.

[17] Fowler, R.H. and Rushbrooke, G.S. Statistical theory of perfect solutions. *Transactions of the Faraday Society* **33** (1937), pp. 1272–1294.

[18] Garey, M.R. and Johnson, D.S. *Computers and Intractability.* Freeman, San Francisco, 1979.

[19] Gaunt D.S. Exact series-expansion study of the monomer-dimer problem. *Physics Review* **179** (1969), pp. 174–179.

[20] Guggenheim, E.A. *Mixtures.* Clarendon Press, Oxford, 1952.

[21] Hammersley, J.M. Existence theorems and Monte Carlo methods for the monomer dimer problem. In *Research papers in statistics: Festschrift for J. Neyman* F.N.David ed., John Wiley, New York, 1966, pp. 125–146.

[22] Hammersley, J.M. An improved lower bound for the multidimensional dimer problem. *Proceedings of the Cambridge Philosophical Society* **64** (1968), pp. 455–463.

[23] Hammersley, J.M. Calculation of lattice statistics. *Proceedings of the Computational Physics Conference* 1970, London, Institute of Physics and the Physical Society, pp. 1–8.

[24] Hammersley, J.M. The number of polygons on a lattice. *2* 7.,73c33,34 **62** (1966), pp. 503–505

[25] Hammersley, J.M., Feuerverger, A., Izenman, A. and Makani, K. Negative finding for the three-dimensional dimer problem. *Journal of Mathematical Physics* **10**(3) (1969), pp. 443–446.

[26] Hammersley, J.M. and Menon, V.V. A lower bound for the monomer-dimer problem. *Journal of the Institute of Mathematics and its Applications* **6** (1970), pp. 341–364.

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

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

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

[30] Heilmann, O.J. and Lieb, E.H. Theory of monomer-dimer systems. *Communications in Mathematical Physics* **25** (1972), pp. 190–232.

[31] Hopcroft, J.E. and Ullman, J.D. *Introduction to Automata Theory, Languages, and Comutation.* Addison-Wesley, Reading MA, 1979.

[32] Jerrum, M.R. Two-dimensional monomer-dimer systems are computationally intractable. *Journal of Statistical Physics* **48** (1987), pp. 121–134.

[33] Jerrum, M.R. An analysis of a Monte Carlo algorithm for estimating the permanent. *Proceedings of the 3rd Conference on Integer Programming and Combinatorial Optimization* (1993), CORE, Louvain-la-Neuve, Belgium, pp. 171–182.

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

[35] Jerrum, M. R. and Sinclair, A. J. Polynomial-time approximation algorithms for the Ising model. *SIAM Journal on Computing* **22** (1993), to appear.

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

[37] Jerrum, M.R. and Vazirani, U.V. A mildly exponential approximation algorithm for the permanent. *Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science* (1992), pp. 320–326.

[38] Karmarkar, N., Karp, R.M., Lipton, R., Lovász, L. and Luby, M. A Monte Carlo algorithm for estimating the permanent. Preprint, 1988. To appear in *Journal of Algorithms.*

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

[40] Kasteleyn, P.W. The statistics of dimers on a lattice I. The number of dimer arrangements on a quadratic lattice. *Physica* **27** (1961), pp. 1209–1225.

[41] Kasteleyn, P.W. Dimer statistics and phase transitions. *Journal of Mathematical Physics* **4** (1963), pp. 287–293.

[42] Kasteleyn, P.W. Graph theory and crystal physics. In *Graph Theory and Theoretical Physics* (F. Harary ed.), Academic Press, London, 1967, pp. 43–110.

[43] Kenyon, C., Randall, D. and Sinclair, A. J. Matchings in lattice graphs. *Proceedings of the 25th Symposium on Theretical Computer Science* (1993), pp. 738-746. (Extended version to appear in *the Journal of Statistical physics* under the title Approximating the number of monomer-dimer coverings of a hyper-rectangular lattice.)

[44] Landau, L.D. and Lifshitz, E.M. *Statistical Physics.* Pergamon Press, New York, 1980.

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

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

[47] Lengauer, T. Algorithmic Research Problems in Molecular Bioinformtics. *Proceedings of the 2nd Israel Symposium on the Theory of Computing and Systems* (1993), pp. 177-192.

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

[49] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A. H. and Teller, E. Equation of state calculations by fast computing machines. *Journal of Chemical Physics* **21** (1953), pp. 1087-1092.

[50] Mihail, M. and Winkler, P. On the number of Eulerian orientations of a graph. *Proceedings of the 3rd ACM-SIAM Symposium on Discrete Algorithms* (1992), pp. 138–145.

[51] Minc, H. *Permanents.* Addison-Wesley, Reading MA, 1978.

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

[53] Onsager, L. Crystal statistics I. A two-dimensional model with an order-disorder transition. *Phys. Rev.* **65** (1944), pp. 117-149.

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

[55] Rasmussen, L.E. Approximating the permanent: A simple approach. *Random Structures and Algorithms* **5** (1994), pp. 349-361.

[56] Roberts, J.K. Some properties of adsorbed films of oxygen on tungsten. *Proceedings of the Royal Society of London A* **152** (1935), pp. 464–480.

[57] Ruelle, D. *Statistical Mechanics: Rigorous Results.* Addison-Wesley, New York, 1989.

[58] Sinclair, A.J. Improved Bounds for Mixing Rates of Markov Chains and Multicommodity Flow. *Combinatorics, Probability and Computing* **1** (1992), pp. 351–370.

[59] Sinclair, A.J. *Algorithms for random generation and counting: a Markov chain approach.* Birkhäuser, Boston, 1993.

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

[61] Sokal, A.D. Monte Carlo methods for the self-avoiding walk. Manuscript (1994).

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

[63] Temperley, H.N.V. and Fisher, M.E. Dimer problem in statistical mechanics—an exact result. *Philosophical Magazine* **6** (1961), pp. 1061–1063.

[64] Valiant, L.G. The complexity of computing the permanent. *Theoretical Computer Science* **8** (1979), pp. 189–201.

[65] Wannier G.H. Antiferromagnetism. The triangular Ising net. *Physics Review* **79** (1950), pp. 357–364.

[66] Welsh, D.J.A. The computational complexity of some classical problems from statistical physics. In *Disorder in Physical Systems* (G. Grimmett and D. Welsh eds.), Clarendon Press, Oxford, 1990, pp. 307-321.

[67] Whittington, S.G. and Soteros, C.E. Lattice animals: rigorous results and wild guesses. In *Disorder in Physical Systems* (G. Grimmett and D. Welsh eds.), Clarendon Press, Oxford, 1990, pp. 323-335.