Approximating the number of monomer-dimer coverings in periodic lattices

Isabel Beichl
National Institute of Standards and Technology, Gaithersburg, Maryland 20899

Dianne P. O'Leary
National Institute of Standards and Technology, Gaithersburg, Maryland 20899
and Computer Science Department and Institute for Advanced Computer Studies, University of Maryland, College Park, Maryland 20742

Francis Sullivan
Institute for Defense Analyses, Center for Computing Sciences, Bowie, Maryland 20715

(Received 22 November 2000; published 18 June 2001)

Our starting point is an algorithm of Kenyon, Randall, and Sinclair, which is built upon the ideas of Jerrum and Sinclair, giving an approximation to crucial parameters of the monomer-dimer covering problem in polynomial time. We make two key improvements to their algorithm: we greatly reduce the number of simulations that must be run by estimating good values of the generating function parameter, and we greatly reduce the number of steps that must be taken in each simulation by aggregating to a simulation with at most five states. The result is an algorithm that is computationally feasible for modestly sized meshes. We use our algorithm on two- and three-dimensional problems, computing approximations to the coefficients of the generating function and some limiting values.

DOI: 10.1103/PhysRevE.64.016701 PACS number(s): 02.70.Uu, 02.50.−r, 05.50.+q

I. INTRODUCTION

We consider the monomer-dimer covering problem for a rectangular lattice with periodic boundary conditions. In three dimensions, for example, the lattice is $n \times m \times p$, with each vertex connected by an edge to its six nearest neighbors. An edge in this lattice corresponds to a possible position for a dimer. A *dimer covering* is a choice of $N = nmp/2$ edges in this graph that defines a complete matching of vertices, i.e., each vertex in the graph is contained in exactly one edge of a dimer covering. A *monomer-dimer covering* is a selection of fewer than N edges in which each vertex is contained in at most one edge; the vertices not contained in any edge are termed monomers.

In this work, we concentrate on two- and three-dimensional lattices but the algorithms are applicable to other dimensions. For simplicity of notation we assume $m = n$ and, in three dimensions, $p = n$, but this is not essential to the algorithms.

Our starting point is an algorithm of Kenyon, Randall, and Sinclair (KRS) [1], which is built upon the ideas of Jerrum and Sinclair [2], giving an approximation to crucial parameters of the covering problem in polynomial time. We make two key improvements to their algorithm: we greatly reduce the number of simulations that must be run and the number of steps that must be taken in each simulation. The result is an algorithm that is computationally feasible for modestly sized meshes.

In Sec. II we give some background on this problem and in Sec. III we summarize the algorithm of Kenyon, Randall, and Sinclair. Section IV presents the improved algorithm A-PRE, (aggregation-prediction) explaining how to reduce the number of simulations. In Sec. V we make the key observation that reduces the number of steps.

We conclude with some experimental results in Sec. VI and some final remarks in Sec. VII.

II. BACKGROUND

This work is an instance of the general technique of restating questions in statistical physics as combinatorial counting problems for which reasonably efficient Monte Carlo approximation techniques may exist. The monomer-dimer problem has its origin in crystal physics where it has been used to model behavior of systems of diatomic molecules (''dimers'') adsorbed on the surface of a crystal. In the three-dimensional case, it occurs in the theory of mixtures and in the cell-cluster theory of liquids. A classical reference is Kastelyn [3]. More general dimer counting questions occur in formulations of the Ising problem as counting matchings on a decorated lattice. A recent theoretical treatment for the Ising model is given by Regge and Zecchina [4]

One of our principal results is based on the idea of ''aggregating'' a Markov chain to one having many fewer aggregated states [5,6]. In our case, we use aggregation to estimate how many steps to wait between collecting samples in the original chain and where to place the maximum of the probability distribution function. This has some similarities with the notion of multicanonical Monte Carlo [7].

III. THE KRS ALGORITHM

Jerrum and Sinclair [2] presented a randomized approximation scheme for approximating the permanent of a 0-1 matrix by studying the behavior of an associated Markov chain. They showed that if the Markov chain had the *rapid* *mixing* property (i.e., within a short time after beginning a random walk, the current state is independent of the initial state), then the approximation could be computed in polynomial time. More precisely, they can approximate the permanent with relative error bounded by ϵ (with high probability) in time that is polynomial in the size of the problem and in $1/\epsilon$. They proposed using this scheme to compute the partition function for the monomer-dimer covering problem.

The ideas of Jerrum and Sinclair were extended by Kenyon, Randall, and Sinclair [1], and their paper is the starting point for our work. They proved that the Jerrum and Sinclair scheme was applicable to a broad class of lattice problems. They gave a proposal for approximating the coefficients of the generating function

$$Z(\mu)=\sum_{s=0}^{N} a_s(N)\mu^s,$$

where $a_s(N)$ is the number of monomer-dimer coverings of the lattice containing exactly s dimers. We will abbreviate $a_s(N)$ by a_s when the size is clear from the context. These coefficients are estimated through knowledge of the equilibrium distribution of a certain Markov chain $\mathcal{M}$ in which adjacent states correspond to monomer-dimer coverings that differ by a single dimer. The equilibrium distribution gives the probabilities of being in a state with i dimers, $i=1,\ldots,N$, after a large number of steps through the chain. The structure of the Markov chain is determined by the lattice of interest, $G=(V,E)$, with $|V|=2N$ vertices in the set V and $|E|=M$ edges in E.

Kenyon, Randall, and Sinclair gather data by taking a random walk controlled by the chain $\mathcal{M}$. In principle, a single Markov chain suffices but in practice they use a sequence of chains, each having the same structure but with transition probabilities depending on a parameter μ, in order to emphasize a certain range of states and get more accurate estimates of their equilibrium probabilities.

For a fixed value of μ, Kenyon, Randall, and Sinclair determine the equilibrium (steady-state) distribution by taking $O(N^2)$ independent samples from $\mathcal{M}$, with $O(\mu' N^2 M)$ steps between sampling where $\mu'=\max(\mu,1)$.

The simulation proceeds as follows:

With probability 0.5, skip this step; otherwise choose, at random, one of the edges.

(1) If the edge can be added to the match, then add it with probability p_{aug}.

(2) If the edge can be deleted from the match, then delete it with probability p_{del}.

(3) If the edge can be swapped into the match by removing some other edge, then swap it.

(4) If the edge can only be added into the match by deleting two other edges, then do nothing.

The probabilities of adding or deleting edges in the match are defined as $p_{aug}=\min(1,\mu)$ and $p_{del}=\min(1,1/\mu)$. Thus, for low values of μ, we are likely to spend most of the time in states with a small number of dimers, but as μ increases, we are more likely to visit states with a large number of dimers.

The device of skipping the step with probability 0.5 is included to permit a bound on the mixing rate of the chain, which, in turn, gives a bound on the number of steps necessary between independent samples [2].

This simulation is run for

$$\mu_1=\frac{1}{2|E|}, \tag{1}$$

$$\mu_i=\left(1+\frac{2}{N}\right)\mu_{i-1}, \quad i=2,3,\ldots, \tag{2}$$

where the largest value of μ should be about a_{N-1}/a_N. From the equilibrium distribution, the probabilities of being in a state with i dimers after a large number of steps through the chain, they calculate ratios $Z(\mu_{i-1})/Z(\mu_i)$, and from these they obtain recursive estimates of the a coefficients.

IV. THE A-PRE ALGORITHM

The KRS algorithm is costly for two reasons.

(1) *The authors determine the number of steps that must be taken between recorded samples, based on the mixing rate of the underlying Markov chain $\mathcal{M}$, to be $O(\mu N^2 M)$.* We show that it is sufficient to base this number on the mixing rate of an aggregated chain $\mathcal{M}_5$.

(2) *The simulation must be run for many values of μ.* The authors note that a sequence of suitable μ values can be determined "by experiment," but we reduce the number of values for μ_i to just N by prediction of the next value based on data gathered for the current value. Our method is in the same spirit as multicanonical Monte Carlo [7].

We will call this algorithm the *aggregation-prediction algorithm*, or the A-PRE algorithm.

We discuss the first cost reduction in Sec. V. To accomplish the second reduction, we use three key relations given in the Kenyon, Randall, and Sinclair paper.

(1) Let $t_i(\hat{\mu})$ be the time that the simulation for $\mu=\hat{\mu}$ spent in states with i dimers. Then $t_i(\hat{\mu})$ is an estimate of the quantity $a_i\hat{\mu}Z(\hat{\mu})$, so the coefficients a_i can be estimated by the recursion

$$a_i=\frac{a_{i-1}\,t_i(\hat{\mu})}{\hat{\mu}\,t_{i-1}(\hat{\mu})}. \tag{3}$$

(2) Although any value of $\hat{\mu}$ suffices, there is less uncertainty in the computed value of a_i if we pick a value of $\hat{\mu}$ for which the simulation spends most of its time in states with i and $i-1$ dimers, and for which the ratio $t_i(\hat{\mu})/t_{i-1}(\hat{\mu})\approx 1$. Thus, from Eq. (3), the ideal value for computing a_i is

$$\hat{\mu}=\mu_i\equiv\frac{a_{i-1}}{a_i}. \tag{4}$$

(3) The proportion of time that the simulation for $\hat{\mu}$ spends in states with s dimers is an unbiased estimator of

$$\frac{a_s\hat{\mu}^s}{Z(\hat{\mu})}.$$

Note that $a_0 = 1$, since there is only one covering with no dimers, and $a_1 = M$. Thus, the optimal value of μ from which to estimate a_1 is

$$\mu_1 = \frac{1}{M}.$$

Using this value of μ, our simulation should spend most of its time in states with 0 or 1 dimers, but it also should spend a significant amount of time in states with 2 dimers, which allows us to estimate the optimal value of μ from which to estimate a_2. Equations (3) and (4) bring us to the formula

$$\mu_{i+1} = \frac{a_i}{a_{i+1}} \approx \frac{a_i}{\dfrac{a_i\, t_{i+1}(\mu_i)}{\mu_i\, t_i(\mu_i)}}$$

so we choose

$$\mu_{i+1} = \mu_i \frac{t_i(\mu_i)}{t_{i+1}(\mu_i)}, \quad i = 1, \ldots, M-1. \tag{5}$$

We could iterate this process to improve μ_{i+1}, but we found that without iteration the ratios $t_i(\mu_{i+1})/t_{i-1}(\mu_{i+1})$ were most often between 0.95 and 1.05.

Thus we can estimate the natural logs of a_i and $Z(\mu_i)$ (since the quantities themselves are too large to store) through the recursions

$$\ln a_i = \ln a_{i-1} + \ln \frac{t_i(\mu_i)}{\mu_i t_{i-1}(\mu_i)}, \tag{6}$$

$$\ln Z(\mu_i) = \ln a_i + i \ln \mu_i - \ln[t_i(\mu_i)/S], \tag{7}$$

where S is the number of states recorded during the simulation.

Using Eq. (2), we see that this drops the number of μ values necessary from

$$\frac{\ln a_{N-1} - \ln a_N + \ln 2 + \ln M}{\ln(1 + 2/N)}$$

to N. This is important since, as Kenyon, Randall, and Sinclair note, ‘‘the running time of the (original) algorithm, though polynomial, is not quite small enough to be genuinely practical.’’

The bookkeeping involved in the simulation is minimal and can be arranged as follows. Color the vertices of the lattice alternately red and black so that no red vertex has a red neighbor and no black vertex has a black one. Then we need to update three arrays: Entries in array *rmatch(i)* record the index of the black vertex matched with the red vertex i, or a zero for unmatched red vertices; entries in array *cmatch(j)* record the index of the red vertex matched with black vertex j, or a zero for unmatched black vertices; the sth entry in array t records the number of times we recorded a state with s matched vertices (dimers).

V. TWO MARKOV PROCESSES: THE ORIGINAL AND THE AGGREGATED

In order to reduce the number of steps between recorded sample values, we need to understand the relationship between two Markov chains.

Let T be the transition probability matrix for the Markov chain $\mathcal{M}$ corresponding to a specific value of μ. Assume that we have ordered the states so that the state corresponding to no dimers is first, followed by the M states corresponding to one dimer, and so forth, up to the states that have N dimers. Note that if we are currently in a state of $\mathcal{M}$ that has k dimers, then at the next time, we must be in a state that has either $k-1$, k, or $k+1$ dimers, Thus the matrix T is block tridiagonal, where the kth block corresponds to states with k dimers:

$$T = \begin{bmatrix} T_{0,0} & T_{0,1} & & & & \\ T_{1,0} & T_{1,1} & T_{1,2} & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & T_{N-1,N-2} & T_{N-1,N-1} & T_{N-1,N} \\ & & & & T_{N,N-1} & T_{N,N} \end{bmatrix}.$$

The block T_{ij} in this matrix contains the transition probabilities from states with j dimers to states with i dimers for $i = 0, \ldots, N$ and $j = i-1, i, i+1$ (as long as these values of j are between 0 and N).

Since T is a probability matrix, all entries are nonnegative and the column sums are one, so that $e^T T = e^T$, where e is the column vector of all 1’s. So e^T is the left eigenvector of T corresponding to the eigenvalue 1, and we will call the normalized right eigenvector p, so that $Tp = p$ and $e^T p = 1$. Thus p is the stationary vector for T, and we will refer to the components corresponding to states with k dimers as the subvector p_k.

Now consider a related but aggregated random process. As we step through the Markov chain $\mathcal{M}$, if we are currently in a state with k dimers, then we will say that we are in state k of the aggregated process. We need to understand the transitions in this aggregated process. In the original process, if we are in the ith state among those with k dimers, then the probability of transitioning to a state with $k-1$ dimers is the sum of the elements in the ith column of the matrix $T_{k-1,k}$. The stationary probability that we are in this ith state is the ith component of p_k. Therefore, the probability of making a transition from a state with $k-1$ dimers to a state with k dimers is $e_{k-1}^T T_{k-1,k} p_k$, where e_{k-1} is a vector of all 1’s, with length equal to the number of rows of $T_{k-1,k}$. Further, the probability of making such a transition, given that we are in a state with k dimers, is $e_{k-1}^T T_{k-1,k} p_k / d_k$, with $d_k = p_k^T e_k$.

Similarly, the probability that the next state also has k dimers is $e_k^T T_{k,k} p_k / d_k$, while the probability of making a transition from a k-dimer state to one with $k+1$ dimers is $e_{k+1}^T T_{k+1,k} p_k / d_k$.

Thus, the transition matrix for the aggregated process $\mathcal{M}_a$ is

$$T_a = VTP^T D^{-1},$$

where the $(N+1)\times K$ matrix P is defined by

$$P = \begin{bmatrix} p_0^T & & & & & \\ & p_1^T & & & & \\ & & \cdot & & & \\ & & & \cdot & & \\ & & & & p_{N-1}^T & \\ & & & & & p_N^T \end{bmatrix},$$

K is the number of different monomer-dimer coverings, and

$$V = \begin{bmatrix} e_0^T & & & & & \\ & e_1^T & & & & \\ & & \cdot & & & \\ & & & \cdot & & \\ & & & & e_{N-1}^T & \\ & & & & & e_N^T \end{bmatrix}.$$

The matrix $D = PV^T$ is a diagonal matrix, with diagonal entries d_k, and we will refer to the vector with entries d_k as d.

The aggregated process $\mathcal{M}_a$ is also a Markov chain, as we now prove.

Note that T_a is an $(N+1)\times(N+1)$ tridiagonal matrix. This matrix has two interesting properties: first,

$$\begin{aligned}[1,1,\ldots,1]T_a &= [1,1,\ldots,1]VTP^T D^{-1} \\ &= e^T TP^T D^{-1} \\ &= e^T P^T D^{-1} = d^T D^{-1} \\ &= [1,1,\ldots,1]\end{aligned}$$

and second,

$$\begin{aligned}T_a d &= T_a D[1,1,\ldots,1]^T \\ &= VTP^T D^{-1} D[1,1,\ldots,1]^T \\ &= VTP^T [1,1,\ldots,1]^T \\ &= VTp \\ &= Vp \\ &= d.\end{aligned}$$

Thus, since the entries of T_a are non-negative, this matrix is also a probability matrix with stationary vector d for the corresponding Markov chain $\mathcal{M}_a$

Note that the stationary vector d determines the values t_i in Eq. (3) and thus tells us everything we need to know in order to compute the values a and Z.

In fact, the Markov chain $\mathcal{M}_a$ gives us more information than we need, since, for a given value of μ, we need only two entries in d in order to estimate a_i, and then only the following one to estimate the next useful μ value. Therefore, we can aggregate to a five-state Markov chain $\mathcal{M}_5$, in which we gather into a single state those states of $\mathcal{M}_a$ corresponding to fewer than i dimers, and into another state those corresponding to more than $i+2$ dimers. (There are fewer than five states if $i=N-1$ or $i=N$.) The transition matrix T_5 for this chain is also tridiagonal.

Initially we do not know the transition probabilities for T_5, but we can step through the corresponding aggregated chain $\mathcal{M}_5$ by generating random samples using the chain $\mathcal{M}$. The data gathered in this way gives us an approximation to the matrix T_5, and the stationary vector for this chain gives an alternate way to compute a_i.

The time required to estimate the parameters a_s using the A-PRE algorithm depends on the number of samples recorded in each Monte Carlo simulation, and on the number of steps necessary to take between samples to ensure independence.

(1) Number of samples: In estimating a_s we try to choose μ so that the states with cardinality s are maximally likely. Thus, the proportion that we are trying to estimate is at least $(1+N)^{-1}$, ''so by a routine variance calculation a sample of size only $O([N])$ suffices for a good statistical estimate.'' [1]

(2) Number of steps between samples: Jerrum and Sinclair (Ref. [22] in Ref. [1]) have shown that the ''mixing time'' of the original chain is $O(\mu N^2 M)$. Clearly, though, the only relevant parameter is the mixing time of the aggregated chain $\mathcal{M}_5$ and this is determined experimentally by computing the nonunit eigenvalue closest in modulus to 1.

Therefore, each simulation in the A-PRE algorithm requires a manageable number of steps. As a practical matter, we run the simulation with $s=N$ steps between samples, saving the transition probabilities that we observed for each aggregated state. We then compute the subdominant eigenvalue of the estimated transition matrix. We use this to check the value of s. If necessary, we increase the value and rerun.

A serious problem arises with this approach when the grid size becomes large: in single or double precision, the subdominant eigenvalue can be indistinguishable from 1, and in this case the estimated number of steps between samples becomes infinite. There are two possible fixes for this problem.

(1) The (integer) counts that contribute to the entries in our approximation to T_5 are known exactly, as are the normalizing constants that make the column sums equal to 1. Thus we could use symbolic computation to find a rigorous upper bound on the subdominant eigenvalue of our 5×5 matrix, no matter how many digits of it agree with 1. (Note that this a rigorous bound for the eigenvalue of the experimental matrix, which is only an approximation to the transition matrix corresponding to the underlying probabilities.) The bound is computed by symmetrizing the tridiagonal matrix and then using the Sturm sequence properties of the leading principal minors in a bisection algorithm [8].

(2) Kenyon, Randall, and Sinclair proved that the number

TABLE I. Results for two-dimensional lattices.

n	κ	f	α	Time (sec)
4	1.9369	1.4142	82–304	23
6	1.9404	1.3741	176–1291	107
8	1.9369	1.3525	204–4235	302
10	1.9471	1.3568	320–7690	709
12	1.9437	1.3483	417–17049	1408
14	1.9422	1.3442	470–31923	3586

of steps necessary between recorded samples is linear in μ. We could fit a line to the number of steps determined for small values of μ in order to determine the number of steps for large values. We tested this by estimating the values for a $8\times8\times8$ grid for $\mu>1000$ using data for $1\leqslant\mu\leqslant1000$, and the results were quite good, giving neither a gross underestimate nor a gross overestimate.

VI. EXPERIMENTAL RESULTS

A. Experiments on two-dimensional lattices

We ran the A-PRE algorithm on two-dimensional lattices with $n\times n$ vertices and periodic boundary conditions. For each value of μ, the number of moves between recordings α was determined by using n^2 times 1000 steps to estimate the subdominant eigenvalue λ of T_5, and then computing the number for which

$$\lambda^{\alpha}<.05.$$

The total number of recorded moves to determine each set of results in the table was 10 000.

There are two interesting parameters that can judge the effectiveness of these algorithms [9],

$$\kappa=\lim_{n\to\infty}[a_1(n)+\cdots+a_N(n)]^{1/n^2}=1.940\,215\,351\cdots,$$

TABLE II. Results for three-dimensional lattices.

n	κ	λ_n	α	Time (sec)
4	2.1911	0.4734	320–7060	430
6	2.1918	0.4588	708–32270	3815
8	2.1968	0.4546	1.0e+03–4.3e+05	21746

$$f=\lim_{n\to\infty}[a_N(n)]^{1/n^2}=1.338\,515\,152\ldots.$$

Table I reports our computed estimates of κ and f, as well as the range of α and the total execution time in seconds for the algorithm on a Sun Ultra-60.

In contrast, the KRS algorithm skips $O(\mu N^2 M)$ steps between observations. Thus, for example, their α is 2048–27 670 for $n=4$ and 1 075 648–1.34e+09 for $n=14$, assuming that the proportionality constant is 1. In order to get $N^2=196$ recorded steps for $n=14$, they require 8.11e+11 steps, rather than the 7.07e+09 steps we used to get 10 000 recorded steps.

Figure 1 shows the estimated coefficients from the 20×20 simulation.

As in Ref. [10], we fit the data points $\ln a_N/N^2$ to the function $\beta+\gamma/N^2$. The resulting value of β is an estimate of the limiting value of $\ln a_N$. Using the data in the table, plus the value 116.0882 for a 20×20 grid, gives an estimate of 0.2914.

B. Experiments on three-dimensional lattices

We ran the algorithms on three-dimensional lattices with $n\times n\times n$ vertices and periodic boundary conditions. Again, the total number of recorded moves was 10 000.

We compute two parameters [11],

$$\kappa=(a_1+\cdots+a_k)^{1/n^3},$$

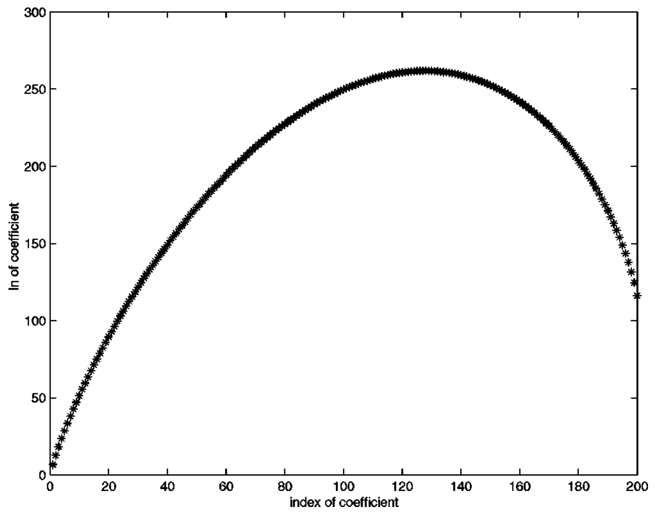

FIG. 1. Estimated coefficients of the generating function for a 20×20 grid.

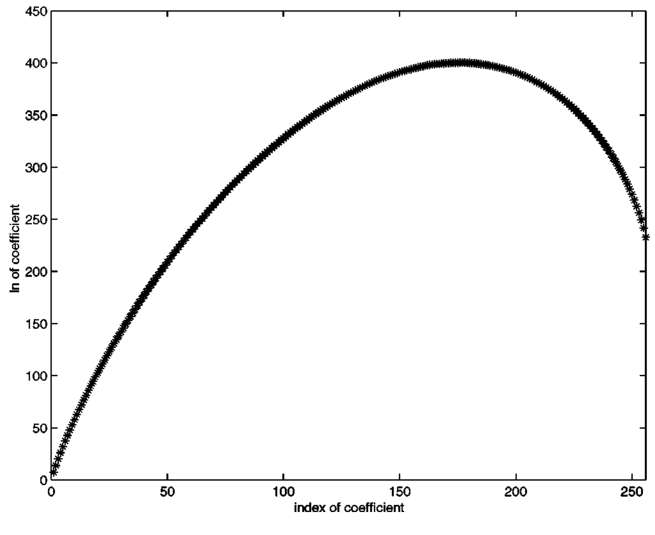

FIG. 2. Estimated coefficients of the generating function for an $8\times8\times8$ grid.

$$0.440\,076 \le \lim_{n\to\infty} \lambda_n \le 0.463\,107,$$

where $\lambda_n = \ln[a_N(n)]/n^3$.

Beichl and Sullivan [10] have estimated the limiting value of λ_n as 0.4466 . . . . Table II shows our results.

Figure 2 shows the estimated coefficients from the $8\times 8 \times 8$ simulation.

Fitting the data points $\ln a_N/N^3$ to the function $\beta + \gamma/N^2$ yields an estimate of 0.4479 for the limiting value.

VII. SOME FINAL COMMENTS

We also experimented with two variants on the algorithm presented in this paper.

(1) More than 50% of the steps in the Kenyon, Randall, and Sinclair are null steps, where we either skip the step entirely, or we decide not to do the add or delete, or there is no possible swap with the chosen edge. We can avoid these null steps by using the Monte Carlo time proposal of Bortz, Kalos, and Lebowitz [12]. Because the A-PRE algorithm requires an estimate of the time spent in various states, it is necessary to keep a running account of the number of Metropolis steps that would have passed between actual moves. This is possible using Monte Carlo time as in [13] and defined in [14]. The values of μ, the number of steps, and the number of steps between recordings are the same as for the KRS Algorithm, but we now keep track of the edges that could be added to the current match and the edges that could be swapped.

(2) Since swaps are quite expensive in the Monte Carlo time algorithm, we are motivated to remove that option and just allow a step in the Markov chain to add or delete an edge. This increases the mixing time of the chain, however, so the number of steps in the simulation increases.

We found that the Monte Carlo time algorithm was not competitive, since its overhead per step was much greater. The ''no swap'' algorithm was not very robust on very small or very large values of μ. Possibly we could use the ''no swap'' algorithm on intermediate values of μ and use swaps on large and small μ, in order to make the no-swap algorithm reliable and perhaps competitive in time with the A-PRE algorithm.

In summary, we have made two key improvements to an algorithm for determining parameters for the monomer-dimer covering problem: we greatly reduced the number of simulations that must be run and the number of steps that must be taken in each simulation. The result is an algorithm that is computationally feasible for modestly sized meshes.

[1] C. Kenyon, D. Randall, and A. Sinclair, J. Stat. Phys. **83**, 637 (1996).

[2] M. Jerrum and A. Sinclair, SIAM J. Comput. **18**, 1149 (1989).

[3] P. W. Kastelyn, in *Graph Theory and Theoretical Physics*, edited by F. Harary (Academic, London, 1967), p. 44.

[4] T. Regge and R. Zecchina, J. Math. Phys. **37**, 2796 (1996).

[5] H. A. Simon and A. Ando, Econometrica **29**, 111 (1961).

[6] P. J. Schweitzer, in *Numerical Methods for Markov Chains*, edited by W. J. Stewart (North-Holland, Amsterdam, 1990), p. 63.

[7] J. Gubernatis and N. Hatano, IEEE Comput. Sci. Eng. **2**, 95 (2000).

[8] J. H. Wilkinson and C. Reinsch, *Handbook for Automatic Computation*, Linear Algebra Vol. II (Springer-Verlag, New York, 1971),

[9] S. Finch, http://www.mathsoft.com/asolve/constant/md/md.html, Technical report, MathSoft Inc., Cambridge, MA, 1999).

[10] I. Beichl and F. Sullivan, J. Comput. Phys. **149**, 128 (1999).

[11] S. Finch, http://www.mathsoft.com/asolve/constant/dmr/dmr.html, Technical report, MathSoft Inc., Cambridge, MA, 1999).

[12] A. B. Bortz, M. H. Kalos, and J. L. Lebowitz, J. Comput. Phys. **17**, 10 (1975).

[13] M. A. Novotny, Comput. Phys. **9**, 46 (1995).

[14] I. Beichl and F. Sullivan, IEEE Comput. Sci. Eng. **4**, 91 (1997).