LBNL-41776
Preprint

ERNEST ORLANDO LAWRENCE
BERKELEY NATIONAL LABORATORY

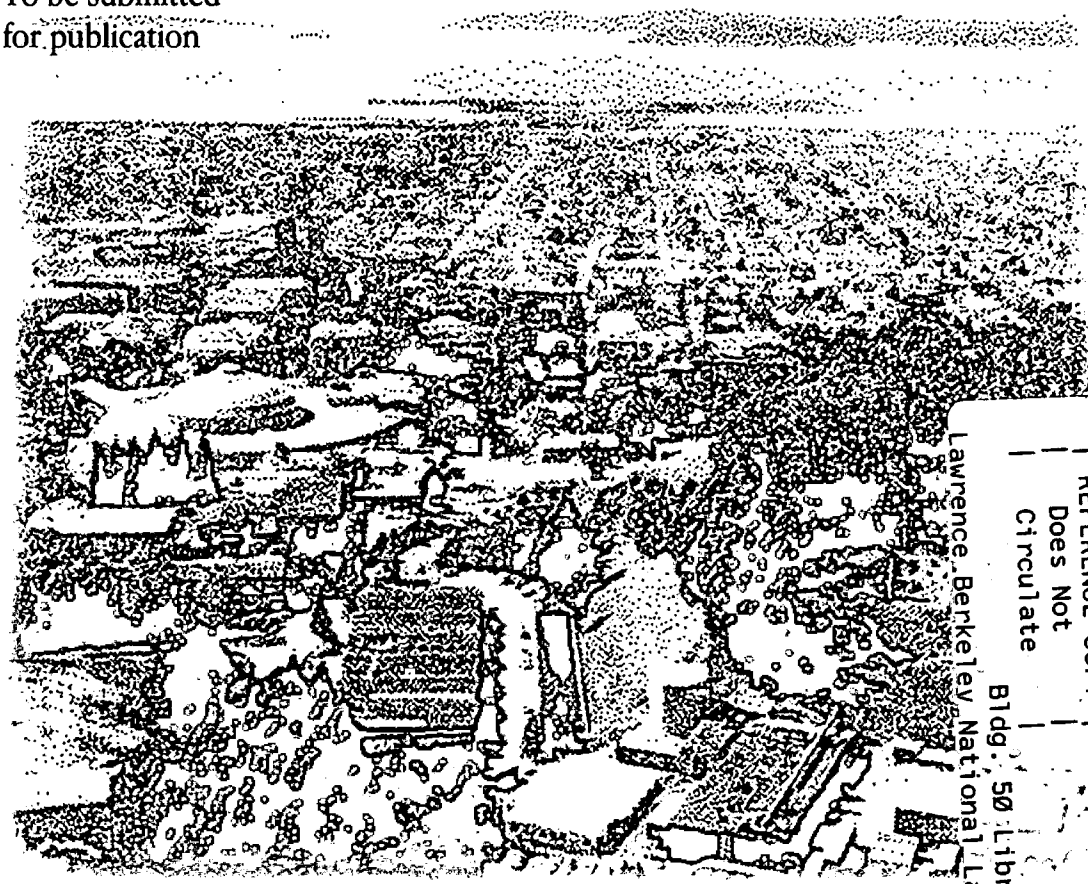
The Applicability of Monte Carlo Methods to Self-Avoiding Walks and Interacting Polymers

Donald M. Krapp, Jr.

Computing Sciences Directorate
Mathematics Department

May 1998

To be submitted
for publication



DISCLAIMER

The applicability of Monte Carlo methods to self-avoiding walks and interacting polymers

Donald Martin Krapp, Jr.*

May 6, 1998

Abstract

We consider polymers, modelled as self-avoiding walks with interactions on a hexagonal lattice, and examine the applicability of certain Monte Carlo methods for estimating their mean properties at equilibrium. Specifically, we use the well-known pivoting algorithm of Madras and Sokal combined with Metropolis rejection to locate the phase transition, which is known to occur at $\beta_{crit} \approx 0.99$, and to recalculate the known value of the critical exponent $\nu \approx 0.58$ of the system for $\beta = \beta_{crit}$. Although the pivoting-Metropolis algorithm works well for short walks ($N < 300$), for larger N the Metropolis criterion combined with the self-avoidance constraint lead to an unacceptably small acceptance fraction. In addition, the algorithm becomes effectively non-ergodic, getting trapped in valleys whose centers are local energy minima in phase space, leading to convergence towards different values of ν. We use a variety of tools, e.g. entropy estimation and histograms, to improve the results for large N, but they are only of

*Supported in part by the Office of Energy Research, Office of Computational and Technology Research, Mathematical, Information, and Computational Sciences Division, Applied Mathematical Sciences Subprogram, of the U.S. Department of Energy, under Contract No. DE-AC03-76SF00098, and by a pre-doctoral Applied Mathematics Fellowship from the National Science Foundation.

limited effectiveness. Our estimate of β_{crit} using smaller values of N is 1.01 ± 0.01, and the estimate for ν at this value of β is 0.59 ± 0.005. We conclude that even a seemingly simple system and a Monte Carlo algorithm which satisfies, in principle, ergodicity and detailed balance conditions, can fail to sample phase space accurately and thus not allow accurate estimations of thermal averages. This should serve as a warning to users of Monte Carlo methods in complicated polymer folding calculations. The structure of the phase space combined with the algorithm itself can lead to surprising behavior, and simply increasing the number of samples in the calculation does not necessarily lead to more accurate results.

1 Introduction

Consider a long linear polymer chain in a good solvent, and suppose the polymer is both dynamically and statically flexible (see de Gennes [5]). When the temperature T is large, the behavior of the polymer is dominated by the excluded volume constraint, and the statistics of such a polymer are known to correspond to those of a self-avoiding walk (SAW). As the temperature is lowered, the attraction between the monomers becomes more important; at a certain critical temperature T_{crit} (called the Θ temperature), the polymer undergoes an abrupt transition from a stretched out chain at $T > T_{crit}$ to a compact blob for $T < T_{crit}$. In addition, there is a third, intermediate configuration at $T = T_{crit}$.

One way to characterize such configurations is by examing the critical exponent ν, which is defined by $< R_N^2 > \propto N^{2\nu}$ where N is the number of monomers in the polymer and $< R_n^2 >$ is the average squared end-to-end distance of polymers composed of N monomers. In two dimensions, the exponent ν depends on the temperature T as follows: for $T < T_{crit}$, $\nu = \frac{1}{2}$; for $T = T_{crit}$, $\nu \approx 0.58$; and for $T > T_{crit}$, $\nu = \frac{3}{4}$. These values of ν are the result of theoretical calculations, numerical work (see below), and experiments conducted on real polymers (see Poole, et al. [12] and Coniglio,

et al. [4]).

As mentioned above, SAW's are appropriate for modeling such polymers when $T > T_{crit}$; however, in general it is necessary to consider interacting SAW's if we wish to model our polymers for arbitrary T. We consider SAW's on a two-dimensional hexagonal lattice with nearest-neighbor (NN) interactions, where an NN interaction occurs when two non-consecutive steps of the walk are one lattice spacing apart. If N_{NN} is the number of such interactions for a given SAW, then the weight of this walk is defined to be $\exp(-\beta N_{NN})$ where $\beta = \frac{1}{T}$ is the inverse temperature. Poole, et al. [12]) used straightforward sampling of SAW's to estimate T_{crit} and the critical exponents at this temperature; their results indicate that for this system $T_{crit} \approx 0.99$.

For a given N and T, in theory it is possible to calculate the exact value of $< R_N^2 >$ for our collection of interacting SAW's; if $\mathcal{S}_N$ is the set of all SAW's of length N, the average is just $\sum_{\alpha \in \mathcal{S}_N} R_N^2(\alpha) \exp(-\beta N_{NN}(\alpha))$. However, since the number of SAW's of a given length N grows exponentially with N, it is impractical to do this for large N. Instead, we use a Monte Carlo technique to generate a sequence (or Markov chain) of walks with large weights; we then calculate an approximation to $< R_N^2 >$ using these walks.

Our algorithm consists of applying the Madras-Sokal pivoting algorithm (see Madras and Sokal [9]) and then using Metropolis rejection (see Metropolis, et al. [11]) to choose those SAW's with high probabilities. We start out with an initial SAW of a given length N; we then randomly pick a step m in the SAW and apply a randomly chosen lattice transformation to the first m steps (if $m < \frac{N}{2}$) or the last $N-m$ steps. If the new walk is not self-avoiding, we reject it and start again; if we get a new SAW, we calculate N_{NN}^{new} and pick a random number $0 \leq r \leq 1$. If $r < \exp(\beta(N_{NN}^{new} - N_{NN}^{old}))$, then the new walk is accepted

For the initialization part, we use thermaliziation: i.e., we discard the walks at the beginning of our sequence before calculating thermal averages. In addition, we use a hash table when calculating the NN interactions, so each step of the algorithm is $O(N)$. Since the algorithm satisfies both the

ergodicity and detailed balance conditions (see Binney, et al. [2]), one would expect that our approximate thermal averages would approach the exact values as the length of the Markov chain increases.

We first attempted to do what we thought would be a relatively simple Monte Carlo calculation of the critical exponent ν as a fuction of the inverse temperature β. The goal was to locate the phase transition which takes place at the Θ point $\beta \approx 0.99$ (see Coniglio, et al. [4] for more details). For $\beta = 0$ all walks have the same probability and there is no nearest-neighbor interaction; the pivoting algorithm at this value of β is very efficient compared with other methods, as is shown in Madras and Sokal [9]. However, we encountered problems when attempting to extend the algorithm to $\beta \neq 0$ which we will explain in the following sections.

2 Acceptance fraction and trapping

For a given iteration of our algorithm, there are three possible outcomes: the new walk is rejected because it is not self-avoiding, the new walk is rejected because its energy is significantly larger than the energy of the current walk, or the new walk is accepted. For a given run, let M be the total number of iterations of our algorithm, and let r_i, r_e, and r_a be the number of walks rejected because of self-interection, the number rejected because of energy considerations, and the number accepted, respectively. We then have $M = r_a + r_e + r_i$.

In Figure 1 we show, as a function of N and β, the fraction of transformations accepted. As N becomes larger, it can be seen that the number of accepted walks becomes smaller: this is to be expected whether β is zero or not. However, the number accepted becomes extremely small as β approaches the critical value $\beta_{crit} \approx 0.99$. At first glance one might think this would have something to do with an increasing number of Metropolis rejections (i.e rejections because of unfavorable changes in the energy). In fact, the fraction of transformations rejected because of energy considerations is

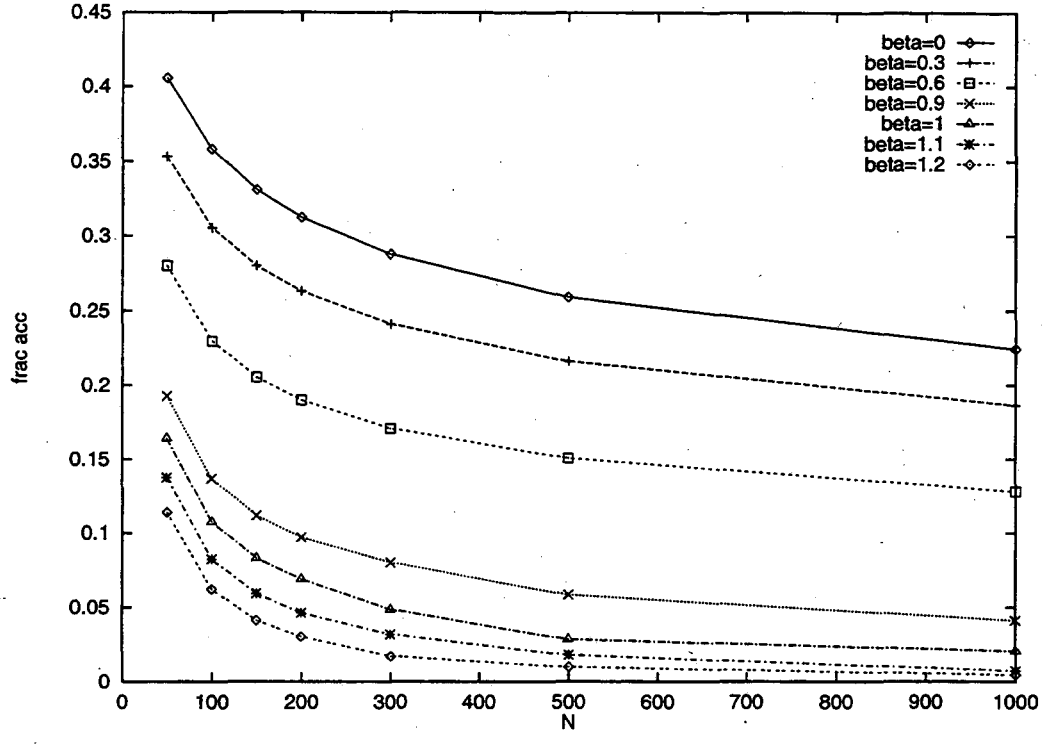


Figure 1: Fraction of attempted transformations accepted vs. length of the walk N for various values of β. For each value of N and β, we first initialized the walks by performing the pivoting algorithm with $T = \infty$ until 10^5 transformations were accepted; we then did then same with $T = \frac{1}{\beta}$. After that, we performed at least 5×10^7 steps of our algorithm and recorded the number of new walks which were both self-avoiding and which were accepted by the Metropolis step. We repeated this whole procedure for five separate runs and averaged the results.

indeed an increasing function of β, although it remains relatively small even near β_{crit}.

In addition, there is also a somewhat less dramatic increase in the percentage of rejections due to self-intersection as β approaches β_{crit}; this can also be attributed indirectly to the Metropolis step, since as β increases, the walks become more contracted, so the probability that a randomly chosen transformation at a randomly chosen step of the walk produces a self-intersecting walk becomes correspondingly greater.

If the rejection percentage increases with N and β, why not just do more iterations of the algorithm? Here we ran into another problem in our calculations which can be termed (computational) non-ergodicity. The non-ergodic behavior can best be illustrated by an example from our data, which is shown in Figure 2. In this case, for $\beta = 0.9$ and $N = 1500$ (a medium-sized walk not too far away from $\beta_{crit} \approx 0.99$) we used the same initial configuration and computed the average end-to-end distance R_{ee} as a function of the length of our Markov chain for ten different runs. For each individual run, we had initially attempted to do a running estimate of the variance of R_{ee} and use this as a stopping criterion: however, we found that in some cases the individual runs were converging to (significantly) different values of R_{ee} and thus gave different values for the critical exponent ν. This behavior is clearly illustrated in Figure 2.

Although we have no direct evidence (one reason is that the size of the phase space for $N = 1500$ is enormous), what we suspect is that the pivoting and Metropolis algorithm becomes computationally non-ergodic for relatively small values of N and values of β near β_{crit}; phase space breaks up into valleys separated by energy peaks which in some cases require an inordinate amount of time to surmount. Although we have a number of walks from each valley, attempting to transform walks from one valley into walks in another in order to illustrate this energy barrier is both difficult and not particularly enlightening. Difficult, because we cannot let the algorithm itself do it: since the sample space is so big, the time required to transform one

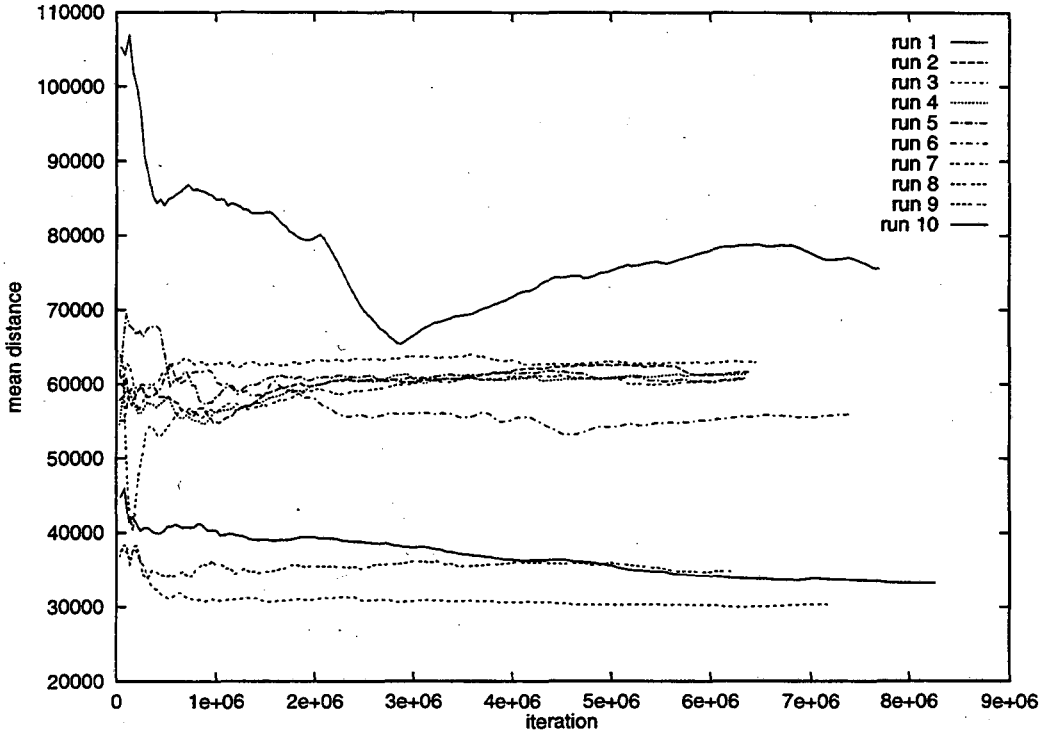


Figure 2: Average end-to-end distance vs. number of iterations for $\beta = 0.9$ and $N = 1500$. In this case, iterations means the number of times we were able to perform a pivoting transformation without self- intersection; the actual number of accepted walks (i.e. after the Metropolis rejection step) was between 1.1×10^6 and 2.5×10^6. See the text for a further discussion of this point. All runs started with the same initial configuration; we first set $\beta = 0.0$ and did 400000 iterations (for $\beta = 0.0$ this means 400000 walks had been accepted), then we set $\beta = 0.9$ and did 400000 more iterations before begining to calculate the end-to-end distance.

randomly selected walk into another is too long; also, developing an algorithm to directly transform one walk into another while maintaing the constraint of self-avoidance is not trivial. Even if we could do the direct transform, and even if the resulting path from one walk to another had a big energy barrier, it would not be a very convincing proof of the existence of the above-mentioned valleys since we would not know if there were any alternative paths between the two walks with lower barriers.

Extrapolation of results for N small can be of some help, but there are a few caveats. Since we can only do the enumeration up to around $N = 30$, it is not clear that conclusions drawn from these phase spaces for such small N will be applicable at much larger N. At these small N the number of nearest-neighbor interactions is so small that β needs to be extremely large in order to allow the formation of valleys and peaks in phase space. For example, we set $\beta = 20$ and then ran our algorithm for small values of N (between 10 and 20); as the reader may have suspected, there was almost immediate convergence towards various walks which represented local minima in the phase space. In Figure 3 we show two such walks for $N = 16$.

A close examination of the two walks in Figure 3 shows that any lattice transformation applied to a part of either of the walks will break some of the bonds (i.e. will decrease the number of nearest-neighbor interactions), and so will be rejected with probability $1 - e^{-\beta}$. If β is large enough, we have from the computational point of view an effectively non-ergodic algorithm; in this case we can define non-ergodic to mean that the number of iterations of the algorithm (and thus the compute time) required to overcome the energy barriers is larger than some given number (say 10^{12}). We suspect that as N increases, the β required for this to occur gradually approaches β_{crit}.

One way of conveying at least heuristically the trapping that we suspect is occuring is to run the algorithm for a fixed N (in this case we use $N = 1000$) and for $\beta = 0$; this gives a sequence of walks for which there is no Metropolis rejection. Since every walk is equally probable, each of the walks in a given series tends to have a different number of nearest-neighbor interactions. We

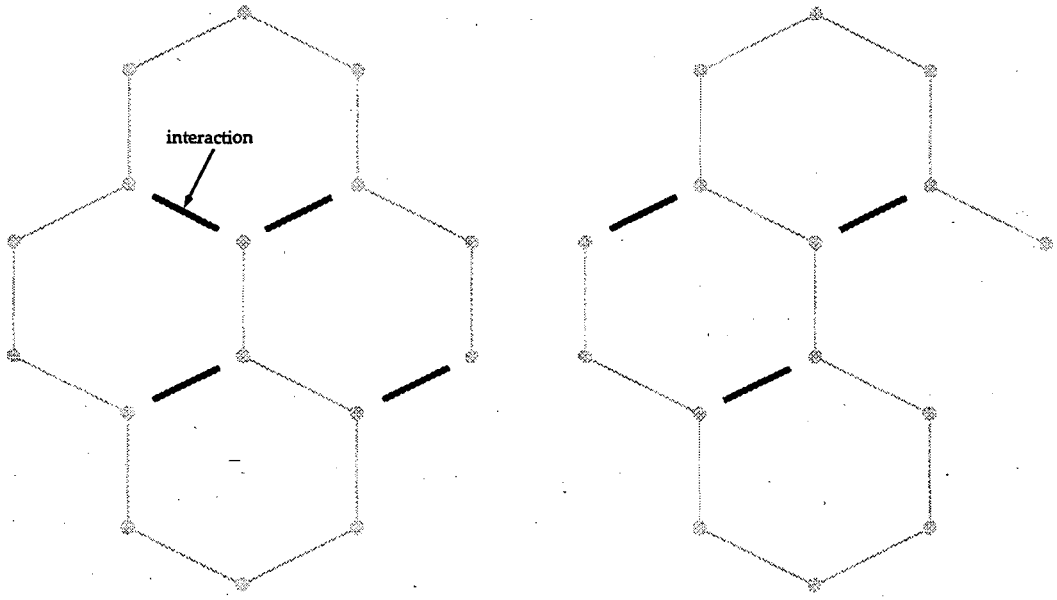


Figure 3: Two "trapped" configurations for $N = 16$.

then set $\beta = 1$ and calculate the probability that the algorithm would actually produce this sequence of walks; the result is shown in Figure 4. Again, we should stress that this is not a proof, but it does give us some insight into what effect an increase in β has on our algorithm's sampling of phase space.

3 Weighting the Runs

The behavior illustrated in Figure 2 occurred for other values of N and other values of β near β_{crit}; convergence towards significantly different values of the mean end-to-end distance is apparently not atypical, so doing one long run is not an effective strategy. Given this, how can we compute the mean with any degree of accuracy? If the phase space breaks up into valleys separated by large energy barriers, then one approach might be to explore a number of different valleys and use the resulting thermal averages from each valley to calculate an overall thermal average.

In other words, we want to do a number of different Monte Carlo runs

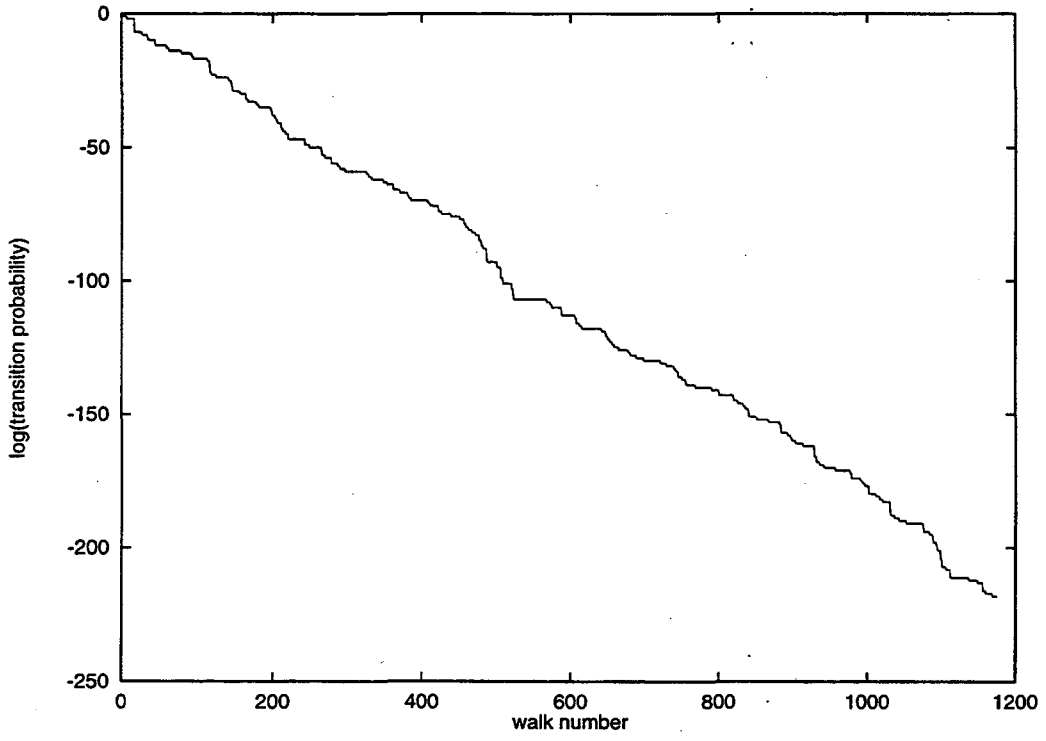


Figure 4: Log(transition probability) vs. walk number for the sequence of walks originally generated using $\beta = 0$. Transition probability means in this case the probability that the Metropolis rejection step would allow us to go from walk 0 to walk i through each of the intermediate walks 1, 2, ..., $i - 1$. When calculating the probabilities, we set $\beta = 1.0$.

and use the means from all of the runs R_{ee}^i to get a final estimate R_{ee} of the mean. However, we have not said how we should weight each individual mean when calculating R_{ee}. There is no reason to suppose that the valleys contain the same number of walks or the same distribution of end-to-end distances, so it is not clear that equally weighting all of the R_{ee}^i is justified.

What is really needed here is a way to estimate how much of phase space is explored by each of the runs; this will allow us to estimate the weights that we should give to the energy valley corresponding to each run. The entropy method (see Meirovitch [10], Chorin [3] and Akao [1]) allows us to quantify how much of phase space is explored: we can use this algorithm to estimate the entropy of each run; the entropy estimate (along with the average energy) can then be used to get an approximate free energy F_i which we can use to get an overall average.

Suppose we have a Markov chain of walks $x_0, x_1, \ldots, x_n$ with each walk consisting of N steps, and suppose we want to calculate the entropy of this chain. Except for very small N, we do not know the partition function and so we cannot calculate the probabilities exactly. Instead, what we do is consider a small portion of each walk, say steps 1 through m where $m \ll N$. We then find all of the self-avoiding walks $y_0, y_1, \ldots, y_l$ of length m, and estimate the probability of each walk by the frequency with which it occurs in our Markov chain; call this $\tilde{P}$. This will give us an estimate of the entropy of this portion of the walks; we then scale the estimate to approximate S for the chain of N-step walks. This gives

$$S = -\frac{N}{m} \sum_{i=0}^{l} \tilde{P}(y_i) \log \tilde{P}(y_i) \tag{1}$$

Now, suppose we have a series of M runs, and denote by r_i the average for the ith run of some quantity r. We can use the entropy estimates to put these averages together as follows. If we know the average energy E_i and average entropy S_i for each run i, we can estimate $< r >$ as

$$< r > \approx \frac{\sum_{i=1}^{M} r_i \exp(-\frac{F_i}{T})}{\sum_{i=1}^{M} \exp(-\frac{F_i}{T})} \tag{2}$$

where $F_i = E_i - TS_i$.

We estimate the entropy for a given segment of the walk and for a given Monte Carlo run in the following manner: let m be the number of steps in the segment. For the first step, let b_{m-1} be 0 if the step is horizontal, 1 if it is in the positive y direction, and 2 otherwise. For the remaining $m-1$ steps, let b_{m-i} be 0 if the step is to the left and 1 if if it is to the right. We thus get a sequence $\{b_i\}$ which uniquely identifies each possible m-step segment. We then form the sum $S_m = \sum_{i=0}^{m-1} 2^i b_i$ which also uniquely identifies each m-step segment; S_m is then stored for each of the walks in our Markov chain. At the end of the run we sort this list, find all of the numbers S_m which appear in the list, and count the number of times that each number appears. This gives us the frequency (i.e. the estimate of the probability) for each configuration, from which we can calculate the entropy estimate using Equation 1.

We attempted to do this for a number of values of β and N, but the results, some of which are illustrated in Figure 5, were disappointing at best. Initially we tried fairly small portions of the walk ($2 \leq m \leq 5$), but we found that the resulting entropies were extremely close to the "equal probability" values for all of the runs: for example, there are $3*2*2 = 12$ walks consisting of three steps, and our entropy values for the three-step segments of the walk were extremely close to $\ln 12$, which is the entropy of the ensemble of equal probability three-step walks. So, we next tried to increase m to between 30 and 50 and then used different segments along the walk to see if the entropy estimation would be the same from segment to segment (a consistency check for the entropy calculation).

However, even for $\beta = 0$ we found that the entropy in the center of the walk was significantly smaller than at the ends. The reason for this is related to one of the improvements in efficiency that we used when implementing the pivoting algorithm: namely, when a given site is selected as the pivot point, we only pivot the smaller part of the walk. Thus, the central part tends to remain in the same configuration for a longer period of time than the ends, so the entropy calculation using the central segment does not accurately reflect

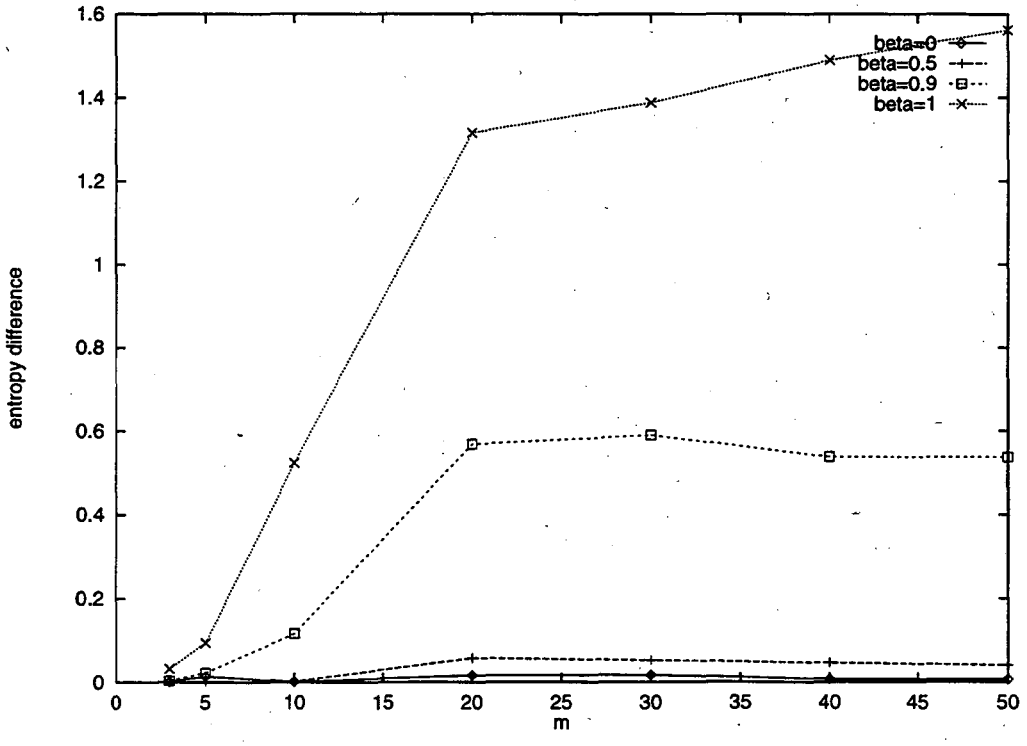


Figure 5: Entropy calculation results for various values of β and m with $N = 1000$; we did 6×10^5 inital iterations, and then calculated the entropy for 2×10^6 iterations of the algorithm. In these cases, the walk segments were $1,\ldots,m$ and $N-m+1,\ldots,N$. All of the estimates start out very close to the "equal probability" value $\ln 12 \approx 2.4849$ of the entropy for $m = 3$. We decided to plot the difference of the two estimates instead of the estimates themselves since for some values of β the estimates are so close that the curves for the two segments overlap.

how much of phase space is explored by the entire walk and leads to an underestimation of the entropy of the walk.

Because of this, we decided to just use segments of length m at the two ends to estimate the entropy. For values of β near 0, the estimates from the different ends were very close; however, there was no need for entropy estimates for these β values since the there was no significant discrepancy between the averages calculated from various Monte Carlo runs. The values of β for which we needed the entropy estimate were precisely those values for which the entropies calculated at the different ends did not agree (see Figure 5).

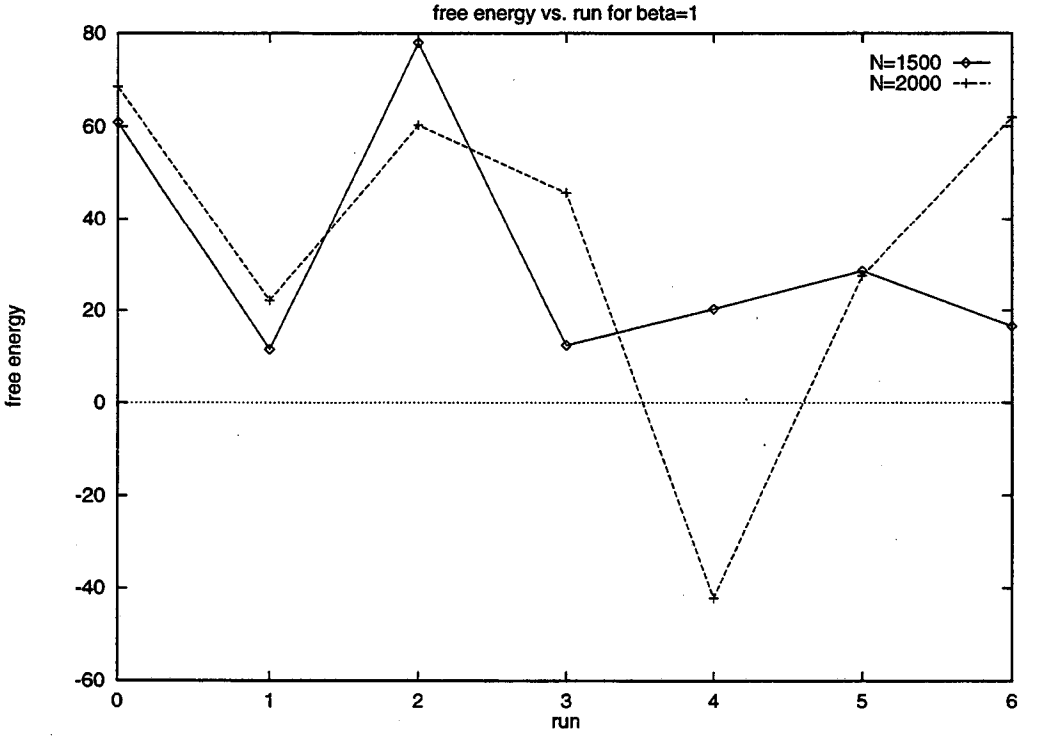


Figure 6: Free energy vs. run for $\beta = 1$, $N = 1500, 2000$.

So, we resorted to the heuristic method of averaging the two entropies to come up with a single estimate; the motivation behind this is that we want the subsection of the walk that we use in estimating the entropy to

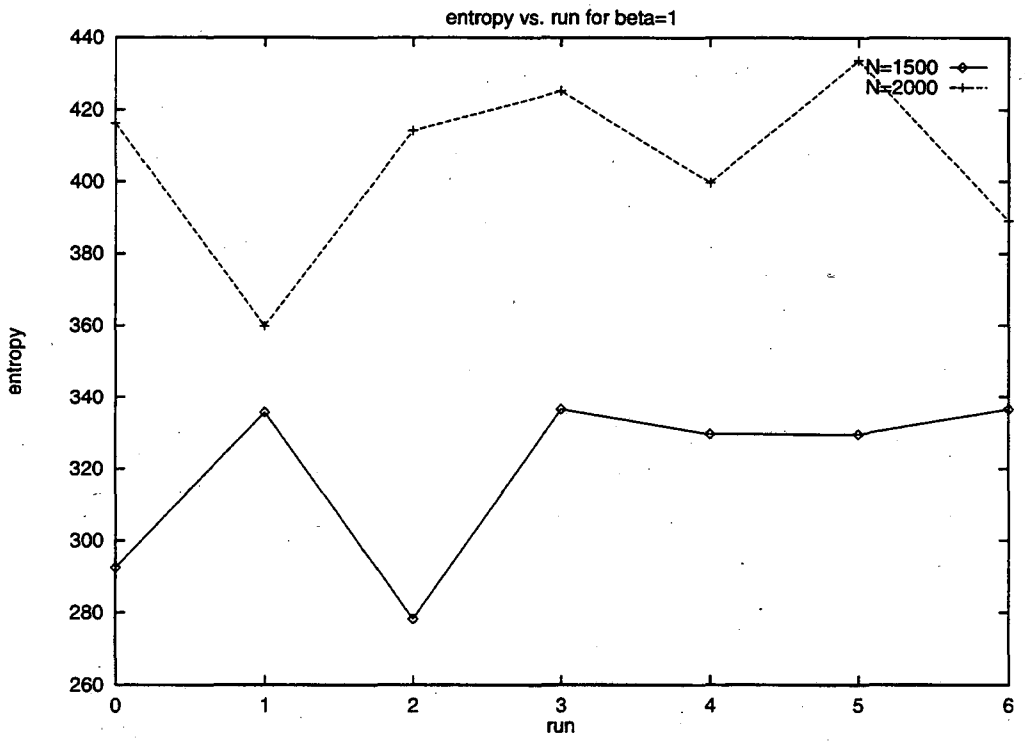


Figure 7: Entropy vs. run for $\beta = 1$, $N = 1500, 2000$.

accurately reflect the entire walk's wandering through phase space. A careful examination of the discussion above about our entropy calculation shows that the arguments there are valid both when we are using consecutive steps of the walk for our subsection and when the "subsection" consists non-adjacent steps of the walk. The only modification necessary is in our function mapping the "subsection" to an integer: if we take non-adjacent steps, in general we would need to do an expansion in base three (instead of base two) since non-adjacent steps could go in any of three directions, whereas adjacent ones are constrained to two because of self-avoidance. Anyway, the point is that by putting the estimates from the two extremes of the walk together, we might hope to get a more global picture of the walk and so a better estimate of the entropy.

After calculating the average entropy, we then tried to use these values to find an average over all of the separate Monte Carlo runs. Once again, the procedure was not effective, and the reason why can be seen if we look at Figure 6 and Figure 7 which show, respectively, the free energy and entropy for a series of runs with β near β_{crit}. What we would expect is that the energy and entropy would be approximately equal for $\beta \approx \beta_{crit}$, since this is where the balancing out between the two takes place. From the graphs, it is clear that for some runs this is indeed (at least approximately) the case; for others, this is not true. If we look at Equation 2, what we might hope is that the free energy for these "bad" runs will be small enough that their contributions to the average will be negligible. However, even if this is true (and it is for this set of runs), the problem is that the same exponential factor that gets rid of these "bad" runs also gets rid of every other run except the one with the largest free energy. The reason for this is that even though the various good runs have free energies which are within a few percent of each other, the actual differences themselves can be of the order of $5 - 10$; when this difference is exponentiated, runs whose free energies are relatively close may have weights whose ratios can be between e^5 and e^{10}. The same phenomenon occurs if we use only the entropies (instead of the free energies) to weight the

walks, although the "chosen" run may be different for the two weightings. All of this is illustrated in Figure 8, where we show the end-to-end distances for the runs as well as the average end-to-end distance computed using the two different weightings.

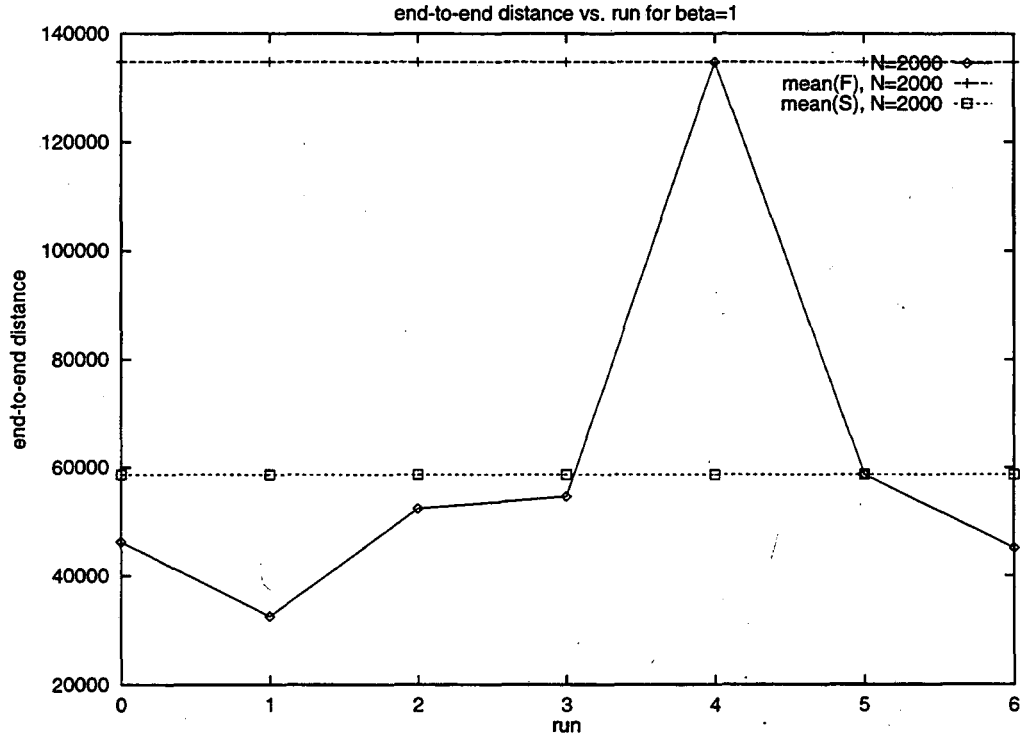


Figure 8: End-to-end distance vs. run for $\beta = 1$, $N = 2000$. mean(F) is the mean calculated using Equation 2. mean(S) is the mean calculated assuming the average energies E_i of each run are roughly the same.

We decided next to temporarily abandon the entropy method and to instead try to look for other ways to weight the averages from the various runs. The first method involved recording the energies of each of the walks in our Markov chain and using this data to construct a histogram for each run; we then lumped all of the histograms from the various runs together to try to get a more accurate average histogram. The resulting histograms for β near β_{crit} and for various N are shown in Figure 9; as can be seen

from the figure, the quality of the composite histograms is unacceptable for even moderate values of N, although for a given N, each of the component histograms may look quite smooth when viewed by itself. For example, the composite histogram for $N = 2000$ appears to be composed of at least four sub-histograms; however, since they are centered around different energies, when put together they do not form a smooth composite histogram.

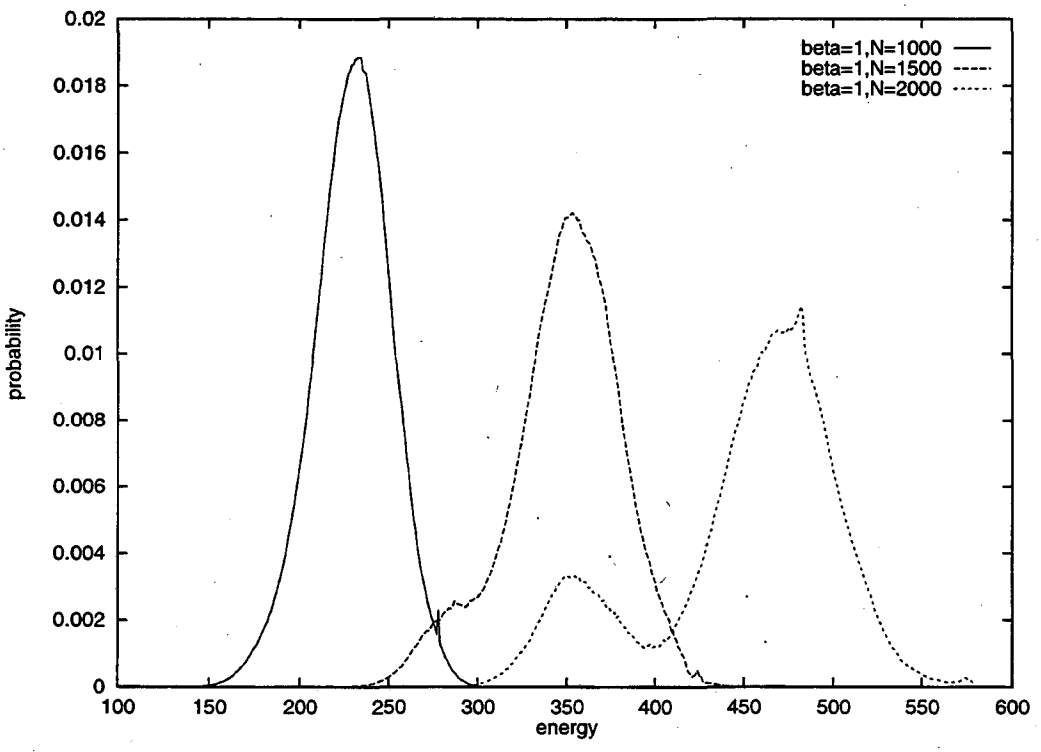


Figure 9: Probability of walks vs. energy (as measured by number of nearest neighbor interactions) for $\beta = 1$ and $N = 1000, 1500, 2000$.

When using the composite histogram method to compute thermal averages, what we are doing is weighting each of the runs according to the walks accepted after the self-avoidance check; what we are not doing is taking into account which runs give us more new walks. In other words, we are only considering the length of each of the separate Markov chains, and not the number of individual walks which each one contains; the reader should recall

that a given Markov chain will contain a sequence of identical walks unless and until the Metropolis step accepts a new walk based on energy considerations. The problem here is that the length of a given Markov chain does not tell us how much this chain has explored phase space. So, instead of considering the length, why do not we consider the number of walks r_a accepted after the Metropolis rejection step? The reader may be saying at this point: what happens if we just keep bouncing back between two (or three or a small number) of walks?

For $N > 100$ the phase space itself is huge; one might suspect that the valleys in which each of the runs get trapped are also huge. What we mean by this is that for any reasonable number of iterations (say $\leq 10^9$), a walk trapped in a subspace only explores a minuscule portion of the subspace. To examine more closely the "hugeness" of the subspaces, and to see if we might be bouncing back and forth between a few walks, we did some test runs for $N = 1000$ and $N = 2000$ with $\beta = 1$; for each walk generated by the algorithm, we recorded the distance between steps i_k and j_k of the walk for $k = 1 \ldots 9$ and $|i_k - j_k|$ varying as $\frac{kN}{9}$. We then used these pairs to label the walks (we didn't have enough disk space to store all of the walks themselves): by examining this list, we found a lower bound for the number of different walks in our chain. For all runs, the average of the ratio of this bound to the number of walks ra accepted by the algorithm is 0.95. In other words, there is very little revisiting of any point in phase space by the algorithm. This does not mean that a given walk does not appear more than one time in our Markov chain (this is not true): what it means is that although we might get stuck at a particular walk for a number of consecutive iterations of the algorithm, once the algorithm accepts another walk, usually we won't go back to the first walk later in the algorithm.

Thus, for a given run i, r_a^i is very close to the number of distinct walks generated by our algorithm. One might suspect that the number of distinct walks would also be directly related to the entropy of a particular run. Since the entropy estimates themselves do not provide an accurate method

of weighting the runs, we might try to somehow weight run i according to r_a^i. In fact, there is a correlation between r_a and the entropy that we computed above. This is illustrated in Figure 10, and this is why we temporarily abandoned the entropy method. Although the fact that we take exponentials of the entropies in order to calculate thermal averages makes these averages inaccurate, the magnitude of the entropies themselves gives us a clue as to which runs we might want to give less weight to and which ones we might want to give larger weights to: since the r_a^i reflect the entropies, instead of using the entropy, why not instead use the r_a^i (which are much easier to compute) to do the weighting?

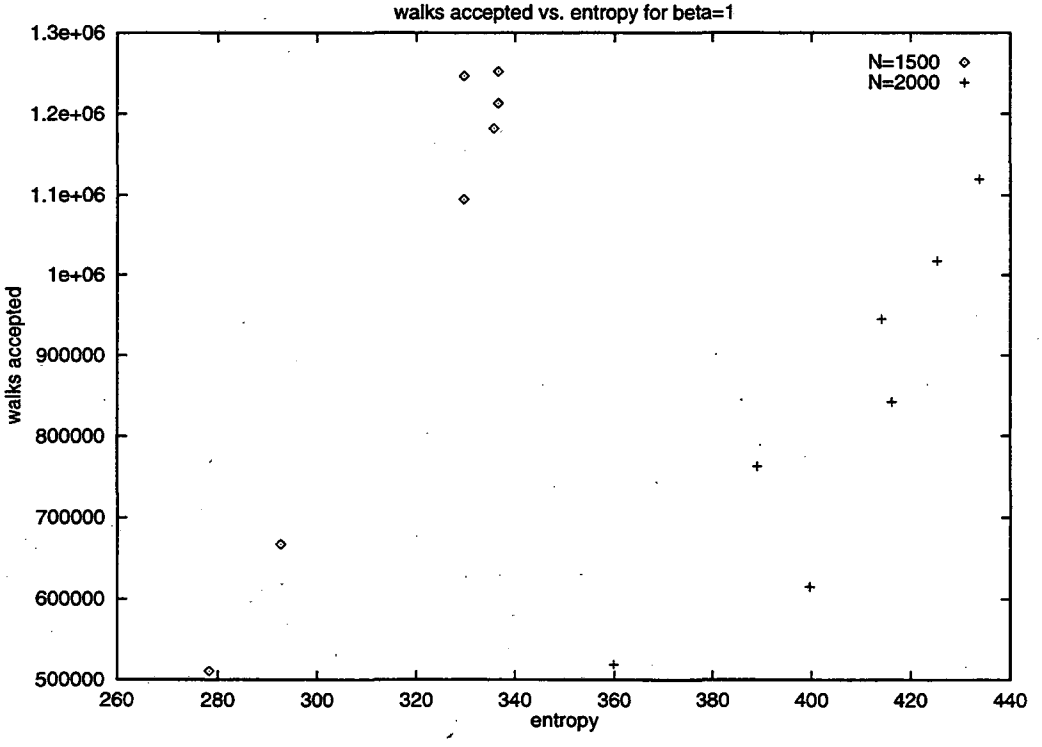


Figure 10: Number of walks accepted after self-avoidance check and Metropolis step vs. entropy for the runs in Figure 6 and Figure 7.

We illustrate the results of this weighting in the following figures. First, Figure 11 shows the results of our calculations when we use no weighting at

all. These are somewhat disappointing, especially given the large number of iterations that we did for each run. For $N > 300$, the curves give inaccurate values of ν near β_{crit}, which is precisely where we need the most accuracy; however, even for smaller N things do not look good near β_{crit}. We also performed the same calculations for larger values of N up to $N = 2000$; the results for these values of N were even more inaccuarate for values of β near β_{crit}. If we examine Figure 12, we see that using the r_a^i gives good results up to much larger values of N, although even this weighting gives inaccurate values of ν for $N > 1000$.

4 Estimating β_{crit} and ν

Even though the curves for large N are not particularly useful, we can still use the curves for small N to try to estimate β_{crit}. To do this, however, we need more sample points near where the curves for different N seem to intersect. We could do many more runs with values of β near the intersection points of the curves in Figure 12, but this would be extremely expensive from a computational point of view. Instead, we decided to use the histogram method to fill in the curves near the suspected value of β_{crit}. The idea behind this is that if we already have the Monte Carlo data for one run at a given β_0, we can use this data for other values of β near β_0: we take the plot of number of walks vs. energy for our run at β_0 and shift this histogram according to the difference between β_0 and our new β (see Akao [1], Kuchta and Etters [8] and Ferrenberg and Swendsen [6] for more details).

Using this method for β_0 and β near $\beta_{crit} \approx 0.99$, we estimate β_{crit} to be between 1.01 and 1.02. Unfortunately, the lack of good data for larger N prevents us from accurately determining the dependance of the estimate of β_{crit} on N.

After having located the approximate value of β_{crit}, we wanted to get a more accurate estimate of the critical exponent ν at this temperature. To do this, we used a procedure from Poole, et al. [12] which goes as follows: if β

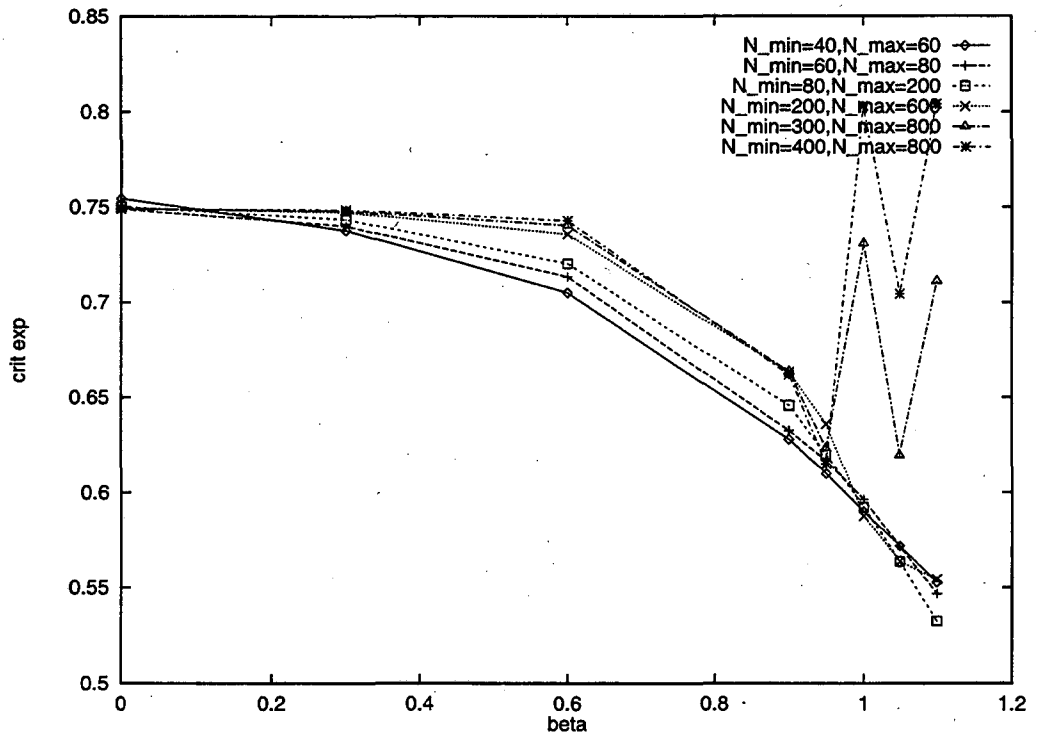


Figure 11: Critical exponent ν vs. β for various values of N. Each pair (β, N) had between 5 and 20 runs; for each run i, we did between 400000 and 800000 inital iterations and then computed the average end-to-end distance $R_{ee}(i, N, \beta)$ for between 3×10^7 and 6×10^7 iterations. In this figure, we combined the $R_{ee}(i, N, \beta)$ without any weighting to get $R_{ee}(N, \beta)$. At every value of β, we then fit a line to the plot of $\log R_{ee}(N, \beta)$ vs. $\log N$ using the set of values of N listed in the graph.

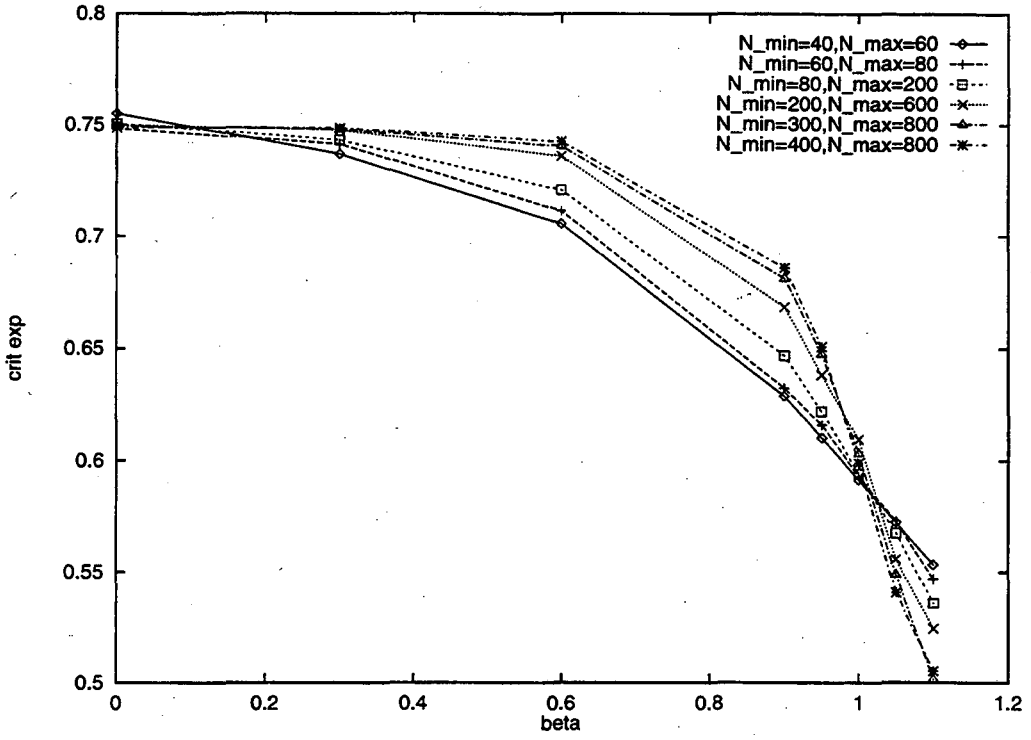


Figure 12: Same as Figure 11, except that when we combined the $R_{ee}(i, N, \beta)$ to get $R_{ee}(N, \beta)$, we first found the maximum ra_{max} of all of the ra_i for each pair (N, β) and eliminated the runs whose ra_i was less than $0.8ra_{max}$; we then used the remaining runs to calculate the average. The reasoning behind this is that if we look back at Figure 10, we see that since r_a^i reflects the entropy S_i, then those runs with small r_a^i and correspondingly small S_i will make a negligable contribution to the overall average. What about weighting the remaining runs after eliminating those with small r_a^i? Well, in this Figure we give them all equal weight, but for our data weighting them using the r_a^i gives almost identical results (recall that the entropy method in effect just selects out one of the runs and gives it a weight disproportionately large compared to those of the other runs).

is fixed, we expect $< R_{ee}^2 >$ to be proportional to $N^{2\nu}$. However, we do not know the constant of proportionality, so we instead take this equation for two different values of N, say N and $N+M$, divide the two equations, take logs, and solve for ν. This gives us $\nu \approx \frac{1}{2} \frac{\log \frac{<Ree(N+M)^2>}{<Ree(N)^2>}}{\log \frac{N+M}{N}}$. We then compute this ratio for various values of N; this gives us several estimates of ν for this particular β. The results of this calculation are shown in Figure 13. For M fixed, the accuracy of this procedure decreases as N increases: this is because we are taking the log of a number which is approaching 1, so any errors in the averages become more and more significant with increasing N. In fact, Poole, et al. [12] use $M = 1$, but we were unable to get any useful results for this M, since for some N we actually had $< R_{ee}(N+1) >$ slightly less than $< R_{ee}(N) >$; the corresponding estimate of ν was therefore less than zero.

As can be seen from Figure 13, for the estimated value of $\beta_{crit} \approx 1.01$, $\nu \approx 0.59$, which is in good agreement with the previous estimates of $\nu \approx 0.58$ (see Poole, et al. [12] for a list of these and further references).

5 Conclusions

We originally attempted to put together two well-known Monte Carlo algorithms (pivoting and Metropolis rejection) in order to calculate thermal averages for a particular polymer system. The example we considered was at first glance fairly simple: self-avoiding walks on a hexagonal lattice in two dimensions with a nearest-neighbor interaction. Since the thermal averages had already been calculated for this system by other methods, our calculations served as a test for the applicability of the pivoting and Metropolis rejection algorithms to such a polymer system.

Our example contained some of the characteristics of more complicated problems (e.g. polymer folding) that are currently of interest in physics. However, even for our seemingly simple system, we ran into a number of difficulties such as unacceptably small acceptance percentages and trapping in valleys centered around local energy minima.

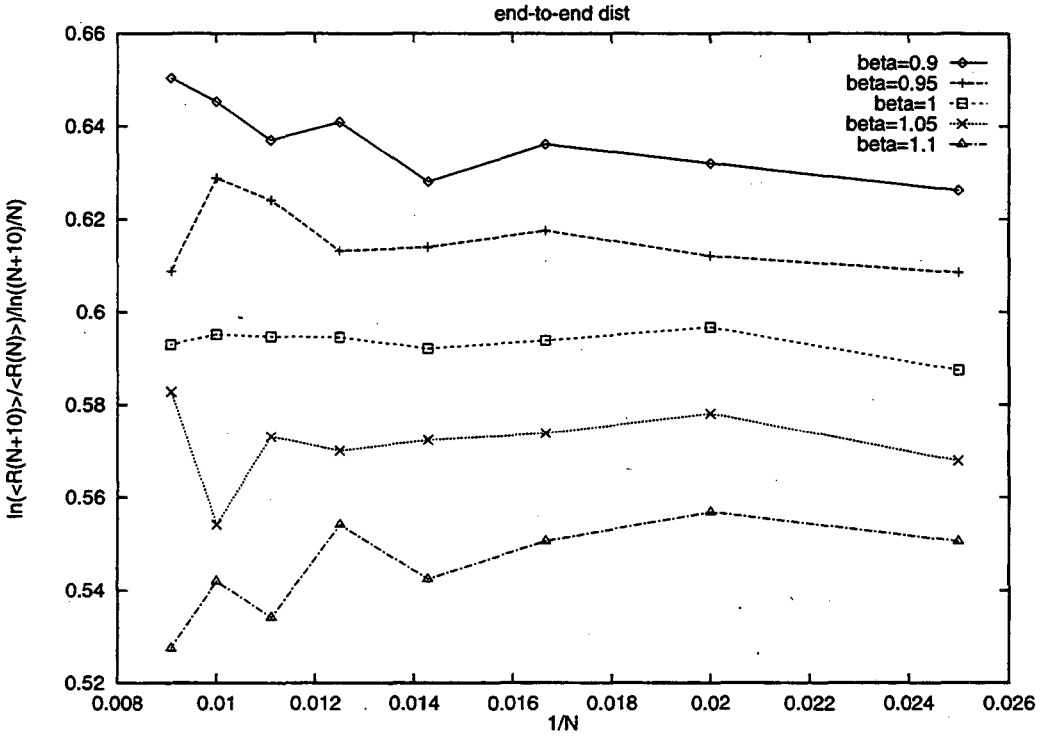


Figure 13: The exponent ν as a function of $\frac{1}{N}$. We set $M = 10$ and tried this for a number of β in the vicinity of our estimate of β_{crit}. Using the radius of gyration instead of $< R_{ee}$ gives almost identical results.

Both of these problems occurred for $N > 800$ (where N is the length of the SAW) and β near $\beta_{crit} = 0.99$. In fact, for any N we needed to concentrate our attention on values of β near the suspected value of β_{crit}. The histogram method was useful for this, since it allowed us to use the data from one run at a given β_0 to estimate the thermal averages for other values of β near β_0; this considerably reduced the number of calculations necessary near β_{crit}. However, the histogram method had its limitations: we could not use it to go too far away from the β_0 at which the original run was done.

The low acceptance percentage mentioned above was caused in part by the length of our walks: even when $\beta = 0$ the pivoting algorithm's acceptance fraction decays as $N^{0.19}$. However, the chief factor was the Metropolis rejection step: when β was near β_{crit}, Metropolis rejection tended to accept walks which had a large number of nearest-neighbor interactions. These walks were contracted, and so a large percentage of the subsequent iterations of the pivoting part of the algorithm led to self-intersecting walks and a correspondingly low acceptance percentage.

To increase the number of acceptances, we could not simply do more iterations. Because the algorithm became trapped in valleys in phase space, for different runs there was a convergence towards different thermal averages (which resulted in different values for the critical exponents). One way out of this difficulty was to do several runs, hoping that in this way we would be able to explore a larger portion of phase space. The problem then arose as to how to put the results from the various runs together.

The entropy method helped us to weight the thermal averages from various runs to produce an overall average. However, it too had its limitations: the entropy estimations gave us an indication of which runs might be more important, but for large enough N, we still could not get accurate values of the exponent ν. This was true even when we tried to put several runs together, each of which had a large number of iterations of the algorithm. In addition, the entropy estimates gave almost all of the weight to the run whose entropy was the highest, and gave very little weight to runs whose

entropy was slightly less than this maximum value.

Nevertheless, the entropy method did prove useful. We discoverd a direct relationship between our entropy estimate for a given run and the number of walks r_a accepted after the Metropolis rejection step. The best results were obtained for relatively small N ($N \leq 800$) by doing several runs and using r_a to weight the averages from the runs. Using these data, our estimates for the critical temperature and critical exponent were $\beta_{crit} = 1.01 \pm 0.01$ and $\nu = 0.59 \pm 0.005$, which are in good agreement with the previous values $\beta_{crit} \approx 0.99$ (based on numerical work by Coniglio, et al. [4]) and $\nu \approx 0.58$ (from theoretical calculations, numerical work, and polymer experiments described in de Gennes [5] and Coniglio, et al. [4]). However, the reason we were able to decide which values of N to use in these estimates of ν and β_{crit} was that we already knew from previous calculations by other researchers what the approximate values of ν and β_{crit} were. Thus, our algorithm was only partially successful, and if we had not known the approximate values of ν and β_{crit} it would have been extremely difficult to decide for which values of N the Monte Carlo method was yielding accurate thermal averages.

What has been learned? Even with simple interactions and a simple and powerful algorithm such as pivoting with Metropolis rejection, it is not necessarily true that one can compute accurate thermal averages for polymers near T_c using Monte Carlo sampling. Researchers who use Monte Carlo in complicated polymer folding calculations should therefore interpret the results of such calculations with care. Simply increasing the number of samples in a Monte Carlo calculation does not always increase the accuracy of the estimates of thermal averages; phase space's structure together with the algorithm itself can lead to complex and unexpected behavior.

6 Acknowledgements

I would like to thank my family, Alexandre Chorin, the National Science Foundation, the Mathematics Department at the University of California at

Berkeley, the members of the Mathematics Department at Lawrence Berkeley National Labs, and the Department of Energy.

This work was supported in part by the Office of Energy Research, Office of Computational and Technology Research, Mathematical, Information, and Computational Sciences Division, Applied Mathematical Sciences Subprogram, of the U.S. Department of Energy, under Contract No. DE-AC03-76SF00098, and by a pre-doctoral Applied Mathematics Fellowship from the National Science Foundation. This work is based upon results published in a Ph.D. thesis [7] in Applied Mathematics at the University of California at Berkeley completed under the supervision of Prof. Alexandre Chorin.

References

[1] J. H. Akao. *Phase Transitions and Connectivity in Three-Dimensional Vortex Equilibria.* Ph.D. thesis, Univ. of California at Berkeley, Berkeley, 1994.

[2] J. J. Binney, N. J. Dowrick, A. J. Fisher, and M. E. J. Newman. *The Theory of Critical Phenomena.* Oxford University Press, New York, 1992.

[3] A. J. Chorin. Equilibrium statistics of a vortex filament with applications. *Commun. Math. Phys.*, 30:619–631, 1991.

[4] A. Coniglio, N. Jan, I. Majid, and H. E. Stanley. Conformation of a polymer chain at the θ' point: Connection to the external perimeter of a percolation cluster. *Phys. Rev. B*, 35:3617, 1987.

[5] P. G. de Gennes. *Scaling Concepts in Polymer Physics.* Cornell University Press, Ithica, NY, 1979.

[6] A. M. Ferrenberg and R. H. Swendsen. Optimized monte carlo data analysis. *Phys. Rev. Lett.*, 63(12):1195–1198, 1988.

[7] D. M. Krapp. *The Applicability of Certain Monte Carlo Methods to the Analysis of Interacting Polymers.* Ph.D. thesis, Univ. of California at Berkeley, Berkeley, 1998.

[8] B. Kuchta and R. D. Etters. Features of the histogram monte carlo method: Application to n2 monolayer melting on graphite. *J. Comp. Phys.*, 108:353–356, 1993.

[9] N. Madras and A. Sokal. The pivot algorithm: a highly efficient monte carlo method for the self-avoiding walk. *J. Stat. Phys.*, 50:109–186, 1988.

[10] H. Meirovitch. A monte carlo study of the entropy, the pressure, and the critical behavior of the hard sphere lattice gas. *J. Stat. Phys.*, 30:681–698, 1984.

[11] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *J. Chem. Phys.*, 21:1087–1092, 1953.

[12] P. H. Poole, A. Coniglio, N. Jan, and H. E. Stanley. Universality classes of the θ and θ' points. *Phys. Rev. B*, 39:495, 1989.

Ernest Orlando Lawrence Berkeley National Laboratory
One Cyclotron Road | Berkeley, California 94720

Prepared for the U.S. Department of Energy under Contract No. DE-AC03-76SF00098