Using path sampling to build better Markovian state models: Predicting the folding rate and mechanism of a tryptophan zipper beta hairpin

Nina Singhal
Department of Computer Science, Stanford University, Stanford, California 94305

Christopher D. Snow and Vijay S. Pande
Department of Chemistry, Stanford University, Stanford, California 94305

(Received 23 December 2003; accepted 19 March 2004)

We propose an efficient method for the prediction of protein folding rate constants and mechanisms. We use molecular dynamics simulation data to build Markovian state models (MSMs), discrete representations of the pathways sampled. Using these MSMs, we can quickly calculate the folding probability (P_{fold}) and mean first passage time of all the sampled points. In addition, we provide techniques for evaluating these values under perturbed conditions without expensive recomputations. To demonstrate this method on a challenging system, we apply these techniques to a two-dimensional model energy landscape and the folding of a tryptophan zipper beta hairpin. © *2004 American Institute of Physics.* [DOI: 10.1063/1.1738647]

I. INTRODUCTION

While experiments can yield a wealth of insight into protein folding, it is difficult for experiments to describe the process of folding in atomic detail. Typically, experiments primarily yield quantitative information on the rate of folding. Ideally, one could use simulation to predict both rate and mechanism of protein folding. The comparison of the rate prediction with experiment could be used as a test of the methodology, and with a validated method, the prediction of the mechanism of folding can yield insight into how proteins fold. Indeed, the direct simulation of protein folding has been a ''grand challenge'' of computational biology for several decades.[1] Simulating protein folding is particularly challenging due to the long time scales involved. While the fastest proteins fold on the microsecond to millisecond time scale, atomistic molecular dynamics simulations are typically constrained to the nanosecond time scale. In order to overcome this fundamental computational barrier, several new computational methods have been proposed.

One such approach to study protein folding events is transition path sampling.[2] Given an initial trajectory between the unfolded and folded regions, this method generates an ensemble of different pathways that join the unfolded and folded regions. From these path ensembles, Bolhuis and coworkers determined the formation order of hydrogen bonds and the hydrophobic core in a β-hairpin.[3] Using the fluctuation–dissipation theorem,[4] it is possible to calculate folding rates from these ensembles.[2] More recently, a new method called transition interface sampling[5] introduced an alternate method to calculate transition rates. One drawback of these methods is that they do not utilize all the simulation results. To ensure that trajectories are decorrelated, only every fifth or tenth pathway generated is added to the path ensemble. Also, these methods require many individual path sampling simulations corresponding to different boundary conditions in order to calculate rates. Since path sampling methods are very computationally demanding, it is interesting to consider whether one can construct an algorithm which can more efficiently utilize simulation data (e.g., folding trajectories) in order to predict folding rates and mechanisms.

There are also techniques that analyze the nature and kinetics of the folding process by representing possible pathways in a graph, or ''roadmap.'' These methods sample configuration space and connect nearby points with weights according to their Monte Carlo probabilities. From these graphs, it is possible to calculate such properties as the shortest path, most probable path, and P_{fold} values,[6] as well as analyze the order in which secondary structures form.[7] The primary challenge of these techniques is the methods used to sample conformational states in order to construct the pathway graph. The graph representation of protein folding pathways does not solve the sampling problem, but recasts it, and sampling any continuous, high dimensional space is still a difficult challenge. Previous graph-based methods have sampled configuration space uniformly (i.e., choosing conformations at random) or used sampling methods biased towards the native state. Clearly, as the protein size increases, it becomes exponentially difficult to sample the biologically important conformations with random sampling. In addition, while probabilistic roadmap methods can predict P_{fold} values[6] and suggest structure formation order,[7] they have not included the time involved in the transitions. Because of this, one cannot predict time dependent properties such as folding rates, and thus it is difficult to assess the experimental validity of these methods.

In this paper, we propose a novel combination of the techniques above. We propose transforming the simulation data gathered from transition path sampling algorithms into a probabilistic roadmap that includes transition time data. As opposed to traditional transition path sampling analysis, this method would incorporate all of the simulated data into the results, therefore potentially yielding an increase in efficiency. We call our model a Markovian state model, or

MSM, as it assumes Markovian transitions between states. From this MSM we can quickly and simultaneously calculate such properties as the P_{fold} for all configurations sampled and the mean first passage time (MFPT) from the unfolded state to the folded state from a single transition path sampling simulation. In addition, this method would provide a compact representation of the possible pathways in the system, which may be useful for understanding the mechanisms involved in folding. We suggest that our method would improve on the current roadmap techniques by sampling points using molecular dynamics, thereby greatly increasing the probability that the configurations that are included are kinetically relevant. In addition, the simulation time between points would inherently capture transition times, making the calculation of folding rates possible.

In the following sections we describe the algorithms necessary to transform molecular trajectories into a MSM with the correct transition probabilities and times. We also provide methods that allow for data gathered at one set of parameters, such as temperature, to be analyzed easily at other parameter values without the need for additional simulations. We then describe analysis techniques to quickly calculate such values as P_{fold} and MFPT. We first give results on a model energy landscape, and find that they are in good agreement with results from direct simulations. Finally, we apply these methods to the analysis of existing simulation data of the folding of a small protein: the tryptophan zipper beta hairpin.[8]

II. METHODS AND THEORY

A. Direct rate calculations

The purpose of this paper is to develop a method for simulating kinetics when one cannot easily directly simulate transitions from one state to another (e.g., for slow transitions from the unfolded state to the folded state). However, to validate the new methods for calculating kinetic properties, it is important to test the methods on systems in which the direct kinetics simulations can be performed. In this case, one can calculate the mean first passage time (in terms of number of Monte Carlo steps for MC simulations and simulated time for Langevin simulations) directly from many independent simulations, even if these simulations are each shorter than the mean folding time.

If one assumes first order kinetics, the probability that a particle has reached the final state at some time t is given by

$$P_f(t) = 1 - e^{-kt},$$

where t is the time, k is the rate, and $P_f(t)$ is the probability of having reached a final state by time t. By running many independent simulations shorter than $1/k$, one can estimate the cumulative distribution $P_f(t)$, and hence fit the value for the rate, k. The mean first passage time is the average time when a particle will first reach the final state, given that it is in an initial state at $t=0$,

$$\text{MFPT} = \int_{t=0}^{\infty} \left(\frac{d}{dt} P_f(t) \right) t \, dt = \int_{t=0}^{\infty} kte^{-kt} \, dt.$$

Integrating by parts yields the solution

$$\text{MFPT} = \frac{1}{k}.$$

One could also find the MFPT by directly calculating the average time when each simulation first reached a final state. However, if some simulations are stopped before the final state is reached because of simulation time constraints, the MFPT calculated will be too low. By first fitting the rate to $P_f(t)$ data (which can be calculated accurately even if some simulations do not finish), one gets a much more accurate MFPT value. For simple systems (such as the two-dimensional energy landscape presented below), one can simply directly simulate kinetics on long timescales.

B. Sampling of paths

For systems where the probability of reaching the final state is very low, the above direct method would require a large number of simulations to get a reasonable estimate of the rate of folding and the mean first passage time. The method that we describe below is a modified version of the shooting algorithm[9] that has been shown to efficiently generate a sample of uncorrelated transition paths leading from the initial region to the final region.

First, we must obtain some initial path between the initial and final regions. This can be obtained from previous data, high temperature unfolding simulations, direct MC or Langevin simulations as above, or some other means. We keep points on this path such that successive points are separated by some time interval, t_{int}. We can label the points along this path as $\{p_0, p_1, \ldots, p_n\}$, where n is the length of the path. We generate new paths by picking a random point along the current path, p_i, and "shooting" a new path from it by starting a new simulation from this point. Points are recorded along this path every t_{int} and are labeled $\{np_0, np_1, \ldots, np_m\}$. If neither the initial nor final state is reached within some simulation time cutoff, we reject this path and the current path remains the same for the next iteration. Otherwise, if either of these states is reached, we stop the simulation at that time point and define our new current path as the combination of the previous current path and the newly generated path as follows. If the new path reached the initial state, then the new current path is $\{np_m, np_{m-1}, \ldots, np_0, p_i, p_{i+1}, \ldots, p_n\}$. If the new path reached the final state, then the new current path is $\{p_0, p_1, \ldots, p_i, np_0, np_1, \ldots, np_m\}$ (Fig. 1). We repeat this shooting step for some set number of trials.

This sampling strategy will capture paths between the boundaries of the initial and final regions. If we are to calculate the MFPT between the initial and final regions, we must also simulate the time a particle can spend within the initial region. To do this, we start many simulations from within the initial region and stop the simulations once the boundary of that region has been crossed.

C. MSM generation

Here, we describe how to generate the MSM of conformational states, including the probability and time to traverse from node to node in the MSM. Each point in the paths

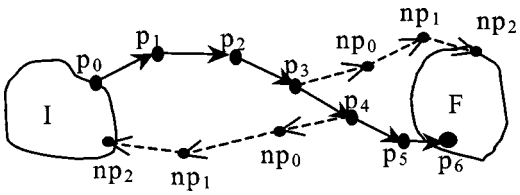

FIG. 1. The shooting algorithm for sampling paths. The solid path shows an original path between the initial and final regions. The dotted paths represent two possible new path segments, corresponding to the new path reaching either the initial or final regions. In the case of the path reaching the initial state, the new path would be $\{np_2, np_1, np_0, p_4, p_5, p_6\}$. In the case of the path reaching the final state, the new path would be $\{p_0, p_1, p_2, p_3, np_0, np_1, np_2\}$.

accepted while sampling paths is represented by a node in the MSM, node_i, for some unique index i. Successive points in each accepted path segment are represented by edges in the MSM, edge_{ij}, representing an edge between node_i and node_j. Each edge has associated with it the simulation time taken to traverse that edge, time_{ij}. Each edge also has associated with it the probability of taking that edge, P_{ij}, which we initialize to one and renormalize in the post-processing step. It is interesting to note that this step may be performed on data generated by the transition path sampling shooting algorithm as above, or on any existing simulation data, so long as the time between points in a simulation is known. It also allows for simulations of different time resolutions to be included in one MSM.

The MSM is designed to embody the possible pathways that the molecule may take while traversing the conformation space. Different paths generated by our simulation methods may pass through very similar conformations, but since the conformation space is continuous, these points will never be exactly the same. However, we wish to capture the fact that these paths reach essentially the same point. We can do this by clustering nearby points in conformation space according to some metric. We define some cutoff value that represents how close two points need to be in order for us to consider them to be the same point. Then, we combine points that are within this distance from one another according to some clustering algorithm. We may choose different cutoffs for the different regions of conformation space, the initial region, the final region, and the transition region. To combine two points, we remove all the incoming and outgoing edges from one of the points and connect them to the other point. If there are now multiple edges between two nodes, we combine them into a single edge with the following values (Fig. 2):

$$P_{ij}^{\text{new}} = P_{ij}^1 + P_{ij}^2,$$

$$\text{time}_{ij}^{\text{new}} = \frac{P_{ij}^1 \,\text{time}_{ij}^1 + P_{ij}^2 \,\text{time}_{ij}^2}{P_{ij}^1 + P_{ij}^2}.$$

The coordinates of clustered points are represented as the weighted average of all points belonging to the cluster.

D. Post-processing of MSM

We need to ensure that every node in the MSM is able to reach a final state. Otherwise, since these nodes will have an infinite mean first passage time, calculations done on the MSM will fail. We identify the nodes that can reach a final state by performing a depth first search from the final states over the incoming edges, and marking all nodes that are reachable.

We propose two different methods for removing the nodes that were not marked. In the first, we simply delete those nodes, thus ensuring that all nodes in the MSM can reach a node in the final state. If there are not many such nodes, this should not bias the results very much. However, if there are many unmarked nodes, deleting these nodes could distort the results. Alternatively, nodes that cannot reach the final state are merged into the closest nodes until all nodes can reach the final state (Fig. 3). This nearest neighbor provides the best guess to the future dynamics of the unmarked node with respect to reaching the final state.

In addition, we normalize the probabilities on all the edges so that on each node, the sum of the probabilities for all outgoing edges is one,

$$P_{ij}^{\text{new}} = \frac{P_{ij}}{\Sigma_{\text{edge}_k} P_{ik}}.$$

The probability on each edge equals the number of times that transition was made divided by the total number of transitions from that node. Given sufficient sampling, these probabilities will converge towards the actual probabilities in the continuous case of each transition from that node.

E. Transition probabilities

The probability of moving between nodes depends on the nature of the dynamics used. Indeed, different transition probabilities can be used to simulate different forms of kinetics from node to node. For example, consider simulations performed using the Metropolis Monte Carlo algorithm to

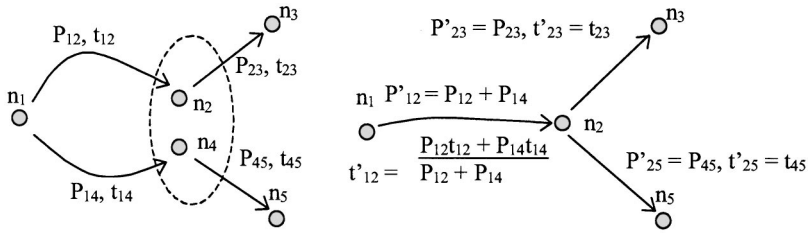

FIG. 2. Clustering of MSM points. If two nodes are closer than a cutoff for some metric, we cluster together these points by replacing them with a new point containing all of their edges. The left picture shows the nodes before clustering, with the dotted circle indicating the nodes that will be merged. The right picture shows the nodes after this clustering step.

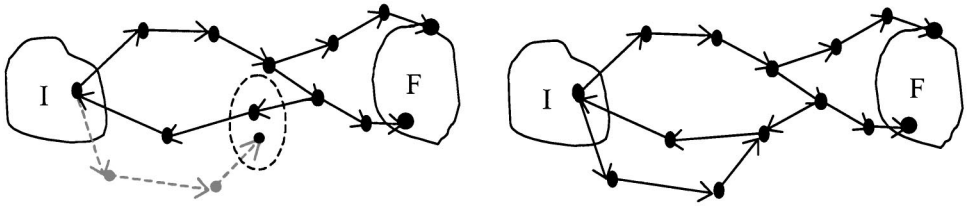

FIG. 3. Clustering of nodes to guarantee that all nodes can reach the final state. In the left picture, the dotted nodes and edges are not able to reach a final state. The dotted circle indicates which two nodes will be merged. The right picture shows the MSM after this step. All nodes can now reach the final state.

generate moves. Given a current point, x, a new point x' is chosen from a distribution $\eta(x,x')$. This move is accepted according to the Metropolis criteria,

$$P_{\text{acc}}=\begin{Bmatrix} 1 & E(x')\leqslant E(x) \\ e^{-[E(x')-E(x)]/kT} & E(x')>E(x) \end{Bmatrix},$$

where $E(x)$ is the energy at point x and T is the temperature.[10]

Langevin dynamics is likely more representative of dynamical properties than Metropolis Monte Carlo, especially since the kinetic interpretation of Monte Carlo relies on the physical nature of the Monte Carlo moves chosen. In Langevin simulations, one performs simulations using the Langevin equation of motion

$$F_{\text{ext}}-m\gamma\frac{dx}{dt}+R=0,$$

$$\langle R(t)R(0)\rangle=2m\gamma k_B T\delta(t),$$

to move particles, where F_{ext} are the external forces acting on the particle, m is the mass, γ is the friction coefficient, and R is a random force, and is assumed to be a Gaussian random variable with the above properties. Rewriting this equation, we can find the change in position with respect to the forces,

$$\Delta x=\frac{R\Delta t}{m\gamma}+\frac{F_{\text{ext}}\Delta t}{m\gamma},$$

$$P\left(\frac{R\Delta t}{m\gamma}\right)=\frac{1}{\sigma\sqrt{2\pi}}e^{-[(R\Delta t/m\gamma)^2/2\sigma^2]},\quad \sigma^2=\frac{2kT\Delta t}{m\gamma},$$

where T is the temperature and the random displacement is distributed according to a normal distribution with standard deviation as above.[11]

We stress that if a Langevin probability is to be used for node transition probabilities, it is imperative that successive nodes be highly related conformationally. Otherwise, if one tries to take large steps (i.e., large Δt), the constant external force approximation will not hold and the transition probabilities become irrelevant. This is the result of over extending the Langevin integrator and it is unclear whether the resulting MSM probabilities will have the desired physical interpretation.

F. Reweighting of edges

The MSM now represents a discrete sampling of the conformation space, and the edges represent the transitions between these states, weighted with the correct probabilities. If some parameters of the system were to change, one could simply adjust the edge weights by the relative probabilities at each value of the parameters to generate a MSM at the new parameters. This assumes that the states and transitions that would be sampled at the new parameters are the same as those sampled at the original parameters. For example, it is common to examine folding at a series of temperatures (T); instead of rerunning the calculation for each temperature, it would be ideal if one could reweight an existing MSM for different temperatures.

This reweighting scheme is loosely analogous to thermodynamic reweighting schemes.[12] While our methodology is for kinetic properties, both methods share the idea of reweighting an ensemble generated at one temperature to yield information at another, and thus both rest on the assumption that the ensemble generated would be useful under the perturbed conditions. Accordingly, one would not expect reasonable results for perturbations that are too large (i.e., temperatures far from the original sampling).

Monte Carlo: The transition between two states as defined by the Metropolis Monte Carlo algorithm is

$$P_{ij}=\eta(\text{node}_i,\text{node}_j)\begin{Bmatrix} 1, & E(\text{node}_j)\leqslant E(\text{node}_i) \\ e^{-[E(\text{node}_j)-E(\text{node}_i)]/kT}, & E(\text{node}_j)>E(\text{node}_i) \end{Bmatrix},$$

where $\eta(\text{node}_i,\text{node}_j)$ is the probability of making a move from node_i to node_j. To reweight the edges at a new temperature, we need the relative probability of each transition at the two temperatures. Dividing the above equation at the two temperatures, we get

$$\frac{P_{ij}(T_1)}{P_{ij}(T_2)} = \frac{\eta(\text{node}_i, \text{node}_j, T_1)}{\eta(\text{node}_i, \text{node}_j, T_2)} \left\{ \begin{array}{ll} 1, & E(\text{node}_j) \leqslant E(\text{node}_i) \\ \dfrac{e^{-[E(\text{node}_j, T_1) - E(\text{node}_i, T_1)]/kT_1}}{e^{-[E(\text{node}_j, T_2) - E(\text{node}_i, T_2)]/kT_2}}, & E(\text{node}_j) > E(\text{node}_i) \end{array} \right\}.$$

Assuming that $\eta(\text{node}_i, \text{node}_j)$ and $E(\text{node}_i)$ are independent of temperature, we get the following equation for the transition probability at the new temperature:

$$P_{ij}(T_2) = \left\{ \begin{array}{ll} P_{ij}(T_1) & E(\text{node}_j) \leqslant E(\text{node}_i) \\ e^{-\Delta E[(1/kT_2) - (1/kT_1)]} P_{ij}(T_1) & E(\text{node}_j) > E(\text{node}_i) \end{array} \right\}.$$

Langevin: The equation of motion for Langevin dynamics is

$$x_{n+1} = x_n + \Delta x_R + \frac{F \Delta t}{m\gamma},$$

where Δt is the time taken to get between the points and Δx_R is distributed according to a normal distribution with mean zero and standard deviation sigma, where

$$\sigma = \sqrt{\frac{2kT\Delta t}{m\gamma}}.$$

We can now define the transition probability between two nodes as the probability of the random displacement needed,

$$\Delta x_R = x(\text{node}_j) - x(\text{node}_i) - \frac{F(x(\text{node}_i))\Delta t}{m\gamma},$$

$$P_{ij} = \prod_{\alpha} \frac{1}{\sigma\sqrt{2\pi}} e^{[-(\Delta x_R^{\alpha})^2/2\sigma^2]},$$

where α represents the dimension of the system. We again wish to compute the relative probabilities at two temperatures, so we divide the above equation at the two temperatures,

$$\frac{P_{ij}(T_1)}{P_{ij}(T_2)} = \frac{\prod_{\alpha} \dfrac{1}{\sigma(T_1)\sqrt{2\pi}} e^{[-(\Delta x_R^{\alpha}(T_1))^2/2\sigma(T_1)^2]}}{\prod_{\alpha} \dfrac{1}{\sigma(T_2)\sqrt{2\pi}} e^{[-(\Delta x_R^{\alpha}(T_2))^2/2\sigma(T_2)^2]}}.$$

Assuming that the forces, mass, and friction coefficient are independent of temperature, and substituting for $\sigma(T)$, we get the following equation:

$$\frac{P_{ij}(T_1)}{P_{ij}(T_2)} = \prod_{\alpha} \sqrt{\frac{T_2}{T_1}} e^{-[(\Delta x_R^{\alpha})^2 m\gamma/4\Delta t][(1/kT_1) - (1/kT_2)]}.$$

Since we know the transition probability at the first temperature from the current MSM, we can calculate the new probability easily at the new temperature using the above equations. We again normalize all edge probabilities so that for each node, the sum of the outgoing probabilities is one. In this way, we generate a MSM at a different temperature without additional simulations.

This analysis can be done on any parameter where it is possible to define the relative transition probabilities in terms of the two parameter values.

G. Mean first passage time and P_{fold} calculation

The MSM consists of a set of nodes and a set of transitions or edges between these nodes. Each edge has a probability associated with it as well as the time taken to traverse this edge. One can define the P_{fold} of a node as the probability that a particle started at that node would reach the final state before reaching the initial state.[13] P_{fold} values have been shown to be useful in understanding the nature of the folding pathway in simplified[13,14] and atomistic[15–17] models. Typically, one calculates P_{fold} values by running multiple simulations (differing by random number seeds or initial velocities) and recording the fraction that fold before they unfold. While this is computationally tractable (compared with a full folding simulation starting from the unfolded state) and naturally parallelizable on massively parallel or grid-computing architectures, it can still be a demanding computational task, especially if the P_{fold} values for many conformations are sought.

Following Apaydin *et al.*,[6] we will use the MSM to calculate P_{fold} values. The P_{fold} can be defined conditionally based on the first transition made from the node,

$$P_{\text{fold}}(\text{node}_i) = \sum_{\text{transition}(i,j)} P(\text{transition}(i,j)) P_{\text{fold}}(\text{node}_i | \text{transition}(i,j)),$$

where the sum is over all possible transitions from node_i. The possible transitions must be mutually exclusive and the sum of their probabilities must be one. The possible transitions from node_i are simply all of the edges leading from node_i, and the probability of each of these transitions is the P_{ij} values defined previously. This satisfies the above condition. $P_{\text{fold}}(\text{node}_i | \text{transition}(i,j))$ is simply the P_{fold} of node_j which results in the following equations:

$$P_{\text{fold}}(\text{node}_i) = \sum_{edge_{ij}} P_{ij} P_{\text{fold}}(\text{node}_j),$$

$$P_{\text{fold}}(\text{node}_i) = 1, \quad \text{node}_i \in F,$$

$$P_{\text{fold}}(\text{node}_i) = 0, \quad \text{node}_i \in I,$$

where I is the initial region and F is the final region. This definition results in a series of n equations for each of the n nodes in the system and n unknowns, the $P_{\text{fold}}(\text{node}_i)$. This system of equations can be solved by iteration as follows:

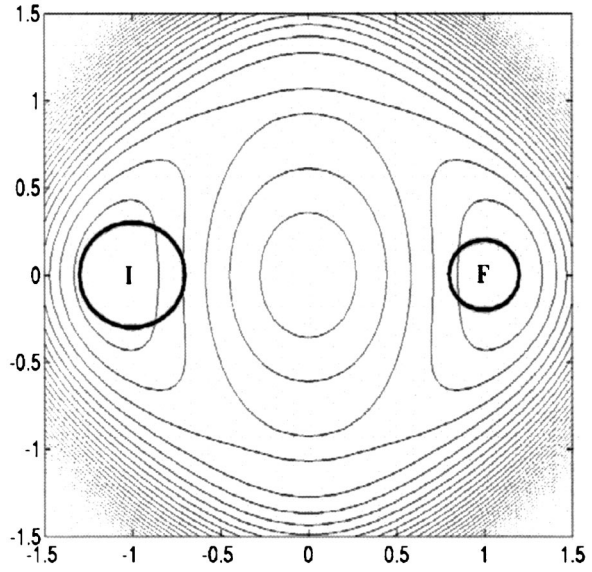

FIG. 4. Contour graph of $E(x,y)$. The initial and final regions are represented by the circles labeled I and F, respectively. The energy difference between the stable regions and the valley in between them is approximately 1 and between the stable regions and the hill in between them is approximately 2 (energy in arbitrary units).

$$\text{Initialize,} \quad \begin{aligned} &P_{\text{fold}}(\text{node}_i)^0 = 1, \quad \text{node}_i \in F, \\ &P_{\text{fold}}(\text{node}_i)^0 = 0, \quad \text{node}_i \notin F; \end{aligned}$$

$$\text{Iterate,} \quad P_{\text{fold}}(\text{node}_i)^{t+1} = \sum_{\text{edge}_{ij}} P_{ij} P_{\text{fold}}(\text{node}_j)^t,$$

until each P_{fold} converges. This iteration method is known as Jacobi iteration.[18] Instead of always using the P_{fold} values from the previous iteration, one can use the new values from this iteration as soon as they become available. This results in the following iterative scheme known as Gauss–Seidel iteration, which converges twice as fast as the Jacobi method,[18]

$$P_{\text{fold}}(\text{node}_i)^{t+1} = \sum_{\text{edge}_{ij}, j \geqslant i} P_{ij} P_{\text{fold}}(\text{node}_j)^t + \sum_{\text{edge}_{ij}, j < i} P_{ij} P_{\text{fold}}(\text{node}_j)^{t+1}.$$

Analogously, it is also interesting to get rate information from simulations. Indeed, rates are a primary mean of comparison to experiment and are thus a critically important quantity to calculate in order to experimentally validate any folding simulation. Rates have not been previously calculated from a roadmap-type representation of states. Below we present a natural generalization of the method to calculate P_{fold} values for the calculation of rates in an efficient and precise manner.

One can define the mean first passage time (MFPT) of any node as the average time taken to get from that node to any node in the final state. The MFPT can be defined conditionally based on the first transition made from the node,

$$\begin{aligned} &\text{MFPT}(\text{node}_i) \\ &= \sum_{\text{transition}(i,j)} P(\text{transition}(i,j)) \\ &\quad \times \text{MFPT}(\text{node}_i | \text{transition}(i,j)), \end{aligned}$$

where the sum is over all possible transitions from node_i. The MFPT of node_i given that a transition to node_j was made is the time it took to get from node_i to node_j added to the MFPT from node_j. This leads to the equation for MFPT of

$$\text{MFPT}(\text{node}_i) = \sum_{\text{edge}_{ij}} P_{ij} \times (\text{time}_{ij} + \text{MFPT}(\text{node}_j)),$$

where the sum is over all edges leading from node_i. In addition, we can define

$$\text{MFPT}(\text{node}_i) = 0, \quad \text{node}_i \in F.$$

This system of linear equations can be iterated in the same way as above except that the initial values for the system should be

$$\text{MFPT}(\text{node}_i) = \infty, \quad \text{node}_i \notin F,$$

$$\text{MFPT}(\text{node}_i) = 0, \quad \text{node}_i \in F.$$

III. RESULTS

A. Model system

We first test the methods outlined above on a simple, two-dimensional model system. Due to their tractability, such model systems are useful for demonstrating the benefits of the proposed method. Our model system is defined by an energy potential of

$$E(x,y) = \frac{(4(1-x^2-y^2)^2 + 2(x^2-2)^2 + ((x+y)^2-1)^2 + ((x-y)^2-1)^2 - 2)}{6}$$

and has been used previously to test transition path sampling methods.[2] The initial and final regions were defined by circles centered at $(-1,0)$ with a radius of 0.2 and $(1,0)$ with a radius of 0.3, respectively. A contour graph of this energy landscape is shown in Fig. 4.

Since this model system is computationally tractable, we can directly compare our proposed methods to direct, brute-force simulations of the kinetics. In particular, we will compare two kinetics methods: Monte Carlo and Langevin dynamics.

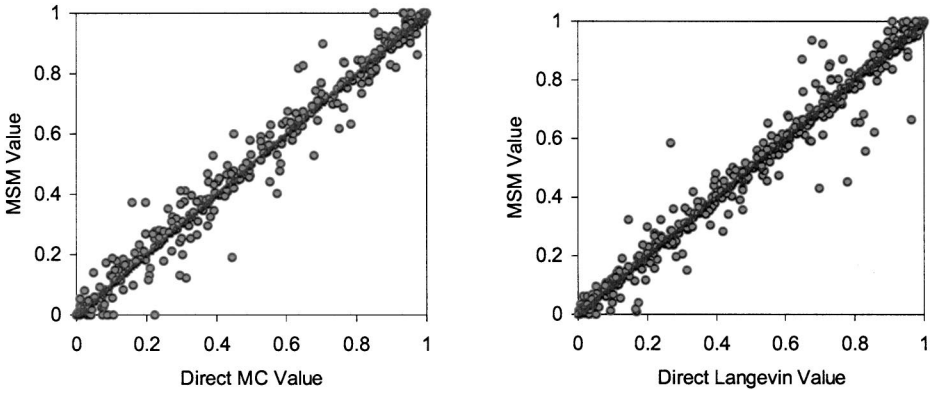

FIG. 5. The correlation between P_{fold} values calculated directly from many simulations and MSM simulations on the same model energy landscape. The left graph shows the comparison for Monte Carlo simulations and the right one shows the same comparison for Langevin simulations.

We performed 10 000 Monte Carlo simulations for each temperature ranging from 0.1 to 1.0, at intervals of 0.1. The move set was defined in each dimension as a normal distribution centered on the current point,

$$\eta(x_\alpha, x'_\alpha) = \frac{1}{\sigma\sqrt{2\pi}} e^{[-(x'_\alpha - x_\alpha)^2/2\sigma^2]}.$$

The standard deviation was defined according to the distance the particle is expected to travel because of diffusion,

$$\sigma = \sqrt{\frac{D\Delta t}{2}},$$

where Δt is the time step and was defined as 0.0001 ps and D is the diffusion constant and equals 91.0 ps^{-1}. In addition, we also ran 10 000 Langevin simulations for each temperature ranging from 0.2 to 1.0, at intervals of 0.1. The forces, F_{ext}, were defined as the gradient of the energy potential given above, the mass m was defined to be 1, and the viscosity γ was defined as 91.0. The time step Δt for these simulations was 1.

For each temperature and type of simulation, five sets of 10 000 independent simulations were started from the initial state, and the time at which they reached the final state was recorded. The initial point in each simulation was sampled randomly from points on the border of the initial region. The mean first passage time for each set was calculated from these 10 000 trials.

MSM generation: To test Monte Carlo kinetics, MSMs were generated on the model energy landscape at temperatures ranging from 0.1 to 1.0, at intervals of 0.1. The time step, Δt, was 0.0001 and the interval at which points on the paths were recorded, t_{int}, was 0.005. Each shooting step was stopped if neither the initial nor final regions were reached after a time of 1.0. Four independent MSMs were generated at each temperature, and each MSM consisted of 10 000 attempted shooting moves. In addition, 50 paths were sampled from the initial state for each MSM. All points in either the initial or final regions were clustered together. For the remaining points, the distance metric chosen was Euclidian distance and the clustering cutoff for each simulation was $\sigma\sqrt{5}$, where sigma is the standard deviation of the normal distribution from which the moves were selected. Points were clustered hierarchically with average-link clustering—the distance between two clusters is equal to the average distance from any member of one cluster to any member of the other cluster. After clustering, any points that could not reach the final state were deleted.

Analogously, Langevin dynamics was examined by generating MSMs at temperatures ranging from 0.2 to 1.0, at intervals of 0.1. The time step was 1.0 and the interval at which points on the paths were recorded was 10.0. Each shooting step was stopped if neither the initial nor final regions were reached after a time of 10 000. Five MSMs were generated at each temperature, and each MSM consisted of 10 000 attempted shooting moves. In addition, 50 paths were sampled from the initial state for each MSM. Clustering was as above with the clustering cutoff as $\sigma\sqrt{1.5}$, where sigma is the standard deviation of the normal distribution from which the random component of each move was selected.

P_{fold} comparison: For one MC and Langevin MSM at

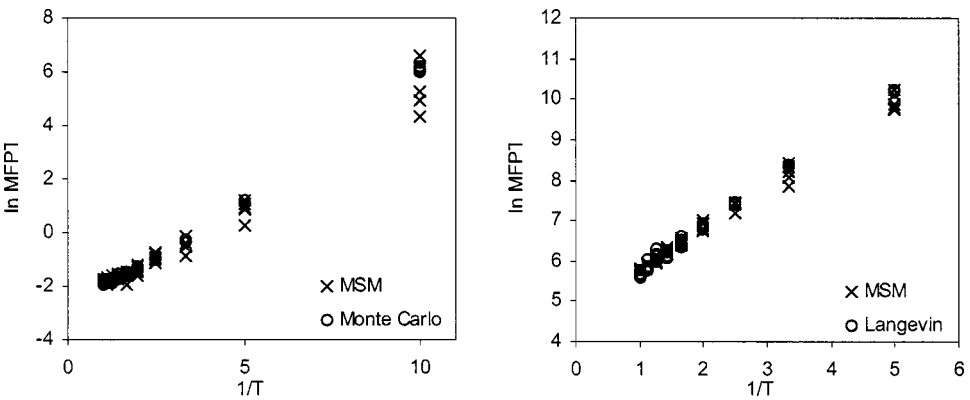

FIG. 6. The comparison between MFPT calculated directly from many simulations and from the MSM simulations as a function of temperature. The left graph shows the result for Monte Carlo simulations and the right one shows the result for Langevin simulations.

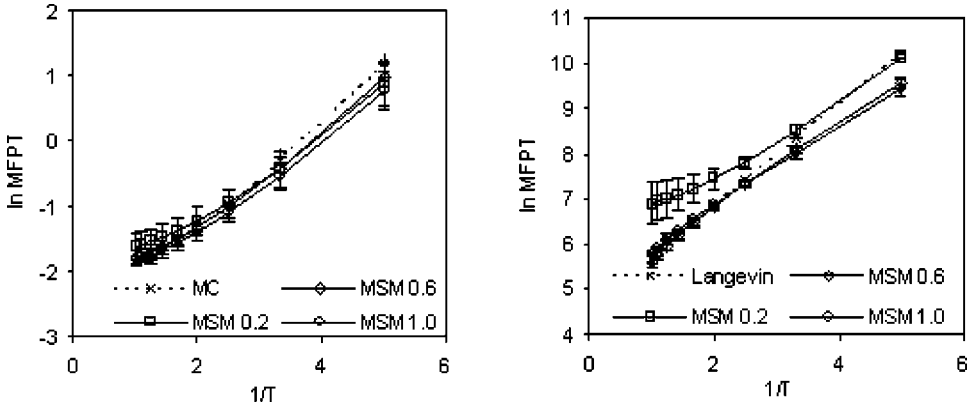

FIG. 7. The comparison between the MFPT calculated from many simulations to the MFPT calculated from reweighted versions of a single MSM as a function of temperature. The square, diamond, and circle points represent the MSMs generated at temperatures of 0.2, 0.6, and 1.0, respectively. The cross points are from the direct MFPT calculations. The left graph shows the results for Monte Carlo simulations and the right graph shows the results for Langevin simulations.

each temperature, the P_{fold} values were calculated for every node. Since it would be too time consuming to compute all P_{fold} values from many direct simulations, about 25–30 nodes were chosen at random from each MSM for comparison. 10 000 MC or Langevin simulations were started at the given temperature from each of these coordinates to compute its P_{fold} value directly. Figure 5 shows the P_{fold} value calculated by many direct simulations compared to those calculated from the MSMs for both simulation types.

The correlation coefficient between the direct MC values and MSM values is 0.989 over all temperature values. For each individual MSM at a given temperature, the correlation coefficient ranges from 0.986 to 0.994. The correlation coefficient between the direct Langevin values and MSM values is 0.990 over all temperatures. The correlation coefficient ranges from 0.986 to 0.999 for each MSM at a given temperature. This shows excellent agreement over a wide range of temperatures for both simulation types.

MFPT comparison: In addition to being able to calculate P_{fold} values at every node, the use of simulation data allows us to estimate the transition times between nodes, and therefore to estimate the MFPT from the initial state. We can compare the MFPTs calculated from the MSMs with the MFPTs we calculated directly from many MC or Langevin simulations (Fig. 6).

The MFPT calculated from the MSMs agree well with those calculated from direct simulations, although the variance among MSM simulations is greater for all temperatures in the MC simulations and for high temperatures in the Langevin simulations.

MFPT from reweighting of edges: We also tested how well our formulation for the reweighting of MSM edges based on temperature was able to predict MFPTs at the new temperatures. For both MC and Langevin dynamics, five additional MSMs were generated at temperatures of 0.2, 0.6, and 1.0. The edges on these MSMs were reweighted to give MSMs at temperatures of 0.2 to 1.0 at an interval of 0.1. The MFPTs calculated from these reweighted MSMs were then compared with those from the direct simulations (Fig. 7).

For the MC simulations, the MFPT calculated from the reweighted MSMs generated at all three temperatures agrees reasonably well over the entire temperature range. The MSMs generated at a temperature of 0.2 show a systematic overestimation of the MFPT for high temperatures. For the Langevin simulations, the MFPT calculated from the reweighted MSMs generated at temperatures of 0.6 and 1.0 also agreed well over the entire temperature range. However, the MFPT calculated from the MSMs generated at a temperature of 0.2 greatly overestimated the MFPT for higher temperatures. At the lower temperatures, we may not be sampling the transitions relevant at the higher temperatures. We examined this possibility by looking at the shortest possible path between the initial and final regions in the MSMs generated at different temperatures. For the Monte Carlo simulations, the MSMs at temperatures of 0.6 and 0.2 showed an increase in the shortest path length of the MSM at a tempera-

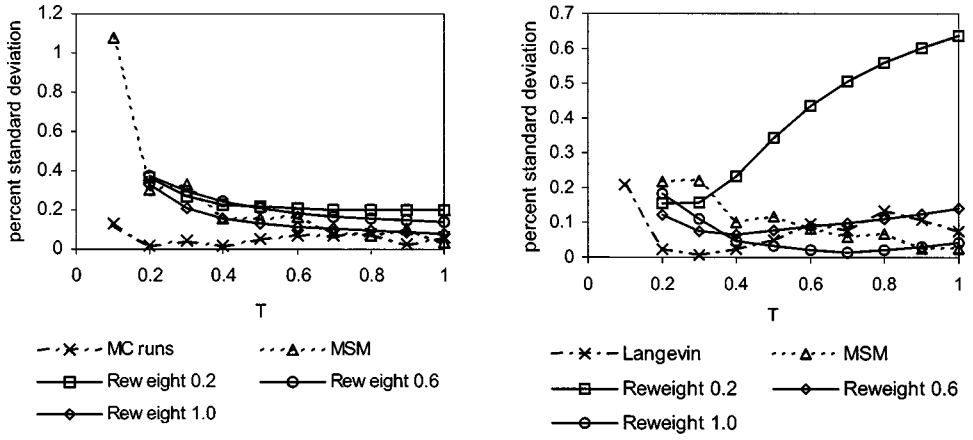

FIG. 8. Error analysis of direct simulations and the various MSM techniques. The graphs show the standard deviation to the average direct simulation value at each temperature divided by that average value. The left graph shows the results for Monte Carlo simulations and the right graph shows the results for Langevin simulations.

ture of 1.0 of 40% and 80%, respectively. For the Langevin simulations, the MSMs at temperatures of 0.6 and 0.2 showed an increase over the shortest path length of the MSM at a temperature of 1.0 of 90% and 300%, respectively. These differences may account for the low temperature MSMs' inability to scale, since we never sample the faster transitions.

To estimate the error in the reweighted MSMs compared to the direct simulations and to MSMs generated individually at each temperature, we compare the MFPT standard deviations at different temperatures for the various methods (Fig. 8). Specifically, we examine the standard deviation relative to the average MFPT at each temperature calculated from the five direct simulations, from the four MSMs generated individually at that temperature, and from the five reweighted MSMs generated at temperatures of 0.2, 0.6, and 1.0 and reweighted to that temperature.

For both the MC and Langevin simulations, the reweighted MSMs have essentially the same error as the MSMs generated individually at each temperature except for the Langevin MSM generated at a temperature of 0.2. At low temperatures, the MFPT calculated directly from simulations has a percent standard deviation which is approximately half that of the MSMs. However, at higher temperatures, the error between the MFPT calculated directly from simulations and that from MSMs is only slightly lower for MC simulations and higher for Langevin simulations.

B. TrpZip kinetics

In addition to the model energy landscape, we applied our methods above to a small protein, the 12-residue tryptophan zipper beta hairpin, TZ2.[19] TZ2 has previously been simulated on Folding@Home.[20] Our goal here is to use these trajectories from Folding@Home to build a MSM to further study the folding of TZ2. This should be a much more challenging test of our methods than the simple two-dimensional example above. If successful, we propose that MSM-based methods would allow one to extend the Folding@Home distributed computing methods to examine the folding of slower and more complex proteins. Indeed, MSM methods combined with Folding@Home-based sampling may also be able to tackle some fundamental issues in the simulation of protein folding, especially that of proteins with non-single-exponential folding kinetics.[21] The results for TZ2 presented here are intended as a ''proof of concept'' application of our methods to fully atomistic simulation.

Using Folding@Home,[20] TZ2 folding has been simulated using the OPLSaa all atom parameter set[22] and the generalized Born/surface area implicit solvent model[23] at a temperature of 296 K. Trajectories were started from an extended conformation and ranged in length from 10 to 450 nanoseconds. To define the initial and final regions, we used a combination of alpha carbon root mean square distance (rmsd) to the native state (pdb code 1LE1[19]) and hydrogen bond and trp–trp distances. In particular, we track the following set of interatom distances where the number indicates the residue, *n* indicates the backbone amide nitrogen, *o* indicates the backbone carbonyl, and *w* indicates the CD2 side chain atom: $d_1 = n3 - o10 + o3 - n10 + n5 - o8 + o5 - n8 + w2 - w11 + w2 - w9 + w9 - w4$ and $d_2 = d_1 + n1 - o12 + o1 - n12$. The initial region was defined as any configuration that had (rmsd$\geqslant$2.5 or rmsd$+0.125d_1 \geqslant 7.75$) and (rmsd $+0.125d_1 \geqslant 9.5$). The final region was defined as any configuration that had rmsd$<$2.5 and rmsd$+0.125d_1 < 7.75$ and $d_2 < 45$. These cutoffs follow Snow *et al.*, except for the dependence on d_2, which was added to discriminate between native states and a set of frayed nativelike states. For more on the simulation details, see Snow *et al.*[8]

To generate the MSM, we chose a tenth of this data set at random, resulting in 1750 independent trajectories. Of these trajectories, 14 reached the final folded state. Frames from the nonfolding trajectories were selected every 10 ns and frames from the folding trajectories were selected every 250 ps. This was done so that there would be more representative conformations in the transition and final states, while still allowing the number of nodes to stay manageable. As discussed in the MSM generation section, because the edges contain the time taken to traverse them, multiresolution data can be accommodated. This selection of data resulted in a total of approximately 22 400 nodes.

The distance metric for clustering was defined as the root mean square deviation of the inter-heavy-atom distance matrix for two conformations. The clustering was performed hierarchically using the average-link distance as the distance between two clusters. After clustering, nodes which could not reach the final state were merged into their nearest neighbor.

The usefulness of a MSM depends upon the type of ensemble used for its construction (similar to the concept of a basis set). Here, the underlying ensemble is fairly one sided, has relatively few transitions, and is not at equilibrium. Specifically, very few trajectories reached the final state and even fewer unfolded after having folded, so the set of transitions to the initial state was not very well sampled. Accordingly, any P_{fold} values calculated would have been biased since the P_{fold} measures the percentage of trajectories that reach the final state before reaching the initial state. On the other hand, a good estimate of the MFPT was possible since it measures the average time taken for a particle to reach the final state having started in the initial state, which is exactly what our data set represents.

We examined a wide range of clustering cutoffs for both the initial and transitional regions to get an estimate for the MFPT. To compare the effects of clustering cutoff, we also performed a similar experiment on the model energy landscape. We ran 500 MC simulations started from the initial state at a temperature of 0.6 with a time step of 0.0001, total time of 0.1, and recording points every 0.005. Of these trajectories, 135 reached the final state. Again, we varied the clustering cutoffs in the initial and transitional regions.

Holding the transitional region cutoff constant, increasing the initial clustering cutoff causes no change in the MFPT in the model system (Fig. 9). For the TrpZip data, increasing the initial region clustering cutoff causes the MFPT to increase until a cutoff of 2 Å and then plateau. One reason why the TrpZip data shows an initial increase in MFPT that is not seen in the model system data is because the TrpZip conformations are in a much higher dimensional

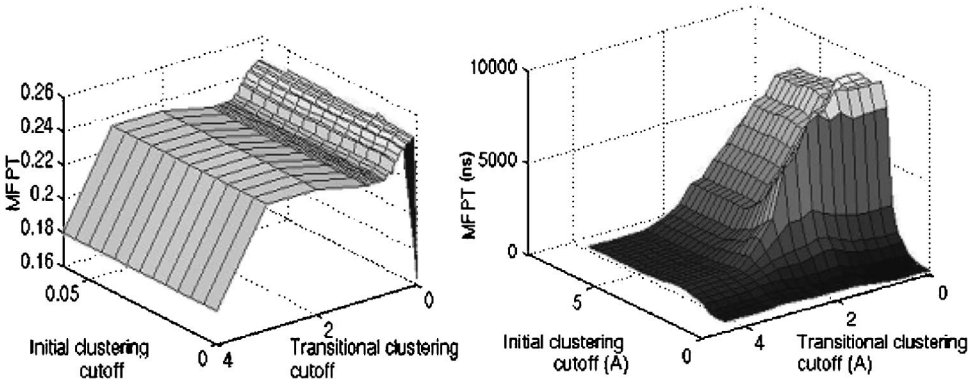

FIG. 9. The effect of clustering cutoff on the calculated MFPT. One axis shows the cutoff in the initial region and the other shows the cutoff in the transitional region. The vertical axis shows the MFPT at each point. The graph on the left is from the model energy landscape and the graph on the right is for the TrpZip protein.

space. If we sample equally in these two spaces, we expect the points in the higher dimensional space to be farther apart. After clustering the two dimensional data to 0.0005, there are 87% fewer nodes. In comparison, after clustering the initial region of the TrpZip data to 2 Å, there are only 76% fewer nodes. There is an increase in the model system data, but only when the transitional cutoff is zero. Holding the initial region cutoff constant and increasing the transitional region cutoff causes the MFPT to decrease in both systems and for all values of the initial cutoff. In the transitional region, we expect that the molecule will go through a series of sequential points on the way to the final state.

The rationale behind clustering is that we can merge together points which are similar by some metric, thus assuming that transitions into or from one of the points is equally likely to go to or come from the other point. This does not apply for points which are sequential along a pathway. Therefore, merging these points causes the MFPT to decrease since we are essentially shortening the length of the path. Because we do not expect any sequential patterns within states in the initial region, increasing the clustering cutoff within this area does not have the same decreasing MFPT as the transitional region.

Over reasonable ranges of cutoffs for the initial (>2 Å) and transitional (1–2.5 Å) regions, we can estimate the MFPT as between 2–9 microseconds. This estimate agrees well with experimental results of 1.8 ± 0.01 μs from fluorescence and 2.47 ± 0.05 μs from IR[8] and with analysis of this simulation data fitting the rate directly of 8 μs for the full data set and 4.5 μs for the random tenth sample used in the MSM analysis.

IV. DISCUSSION AND CONCLUSIONS

We have introduced new computational tools for efficiently analyzing the data collected from Monte Carlo or molecular dynamics simulations. These methods capture the probabilistic and time-dependent nature of the kinetics in a compact Markovian state model representation which can easily be analyzed for properties such as the P_{fold} and MFPT of every node.

The P_{fold} values calculated over a wide range of temperatures and both Monte Carlo and Langevin simulation types show excellent agreement with those calculated by brute force. One area of improvement for this method is that while the MSM values give the same average MFPT as the MFPT calculated from direct simulations, the variance is much higher. This is probably because the MSM simulations are somewhat dependent on the initial path chosen. The only way in which edges can lead from the initial state is either from the initial path or from the sampling of the initial state. Edges resulting from the shooting algorithm all lead into the initial state. One could fix this problem by having many initial paths. However, in a protein example, folding trajectories may be very difficult to generate beforehand. Another way to fix this problem and achieve more precise results would be to include shooting paths which move backwards in time, thus allowing for more edges leading from the initial state. Monte Carlo and Langevin dynamics cannot be run backward in time because the velocity is not maintained between steps. If one were to use some other molecular dynamics simulation system that maintained velocity, then the trajectories may be run backwards in time and this problem could be averted.

In addition to the algorithms necessary to create the MSM from simulation data, we have also described methods for reweighting the edges of the MSM to analyze the system at different parameter values. In particular, we provided transformations for the edge weights at different temperatures, given that the simulations were from either MC or Langevin dynamics. These methods show promise since we can analyze the system at many different parameter values without the need for any additional simulations. The reweighting of the MSMs seemed to work well in general and give results with similar errors as MSMs generated individually at each temperature. The one exception was the Langevin MSMs generated at a temperature of 0.2. These MSMs did not give accurate results when rescaled for temperatures greater than 0.3. One reason for this may be that at the low temperature, the system did not have enough samples in the relevant transitions to scale to higher temperatures. It may be necessary to generate composite MSMs consisting of data from many different temperatures in order to properly rescale to a wider range of temperatures.

This method was then applied to existing folding simulation data from a small β-hairpin protein. We were able to calculate folding rates which were in reasonable agreement with experimental data and previous analysis of the simulation data. The majority of time in these calculations is spent in the clustering step. Currently, we compute the full n^2 distance matrix between all nodes in the MSM, a very expen-

sive calculation. Better clustering algorithms can reduce this computation time.

Future work involves additional analysis of how this procedure can be applied to real protein systems. In particular, we plan to analyze the error in MFPT estimation by looking at different subsets of the entire data set, further explore the effects of the clustering cutoffs, and to characterize folding pathways and mechanisms from the generated MSMs.

ACKNOWLEDGMENTS

N.S. was supported through the National Defense Science and Engineering Graduate Fellowship. C.S. was supported by a predoctoral fellowship from HHMI. The computation was supported by grants from NIH and a gift from Google.

[1] J. E. Shea and C. L. Brooks, Annu. Rev. Phys. Chem. **52**, 499 (2001).
[2] C. Dellago, P. G. Bolhuis, F. S. Csajka, and D. Chandler, J. Chem. Phys. **108**, 1964 (1998).
[3] P. G. Bolhuis, Proc. Natl. Acad. Sci. U.S.A. **100**, 12129 (2003).
[4] D. Chandler, *Introduction to Modern Statistical Mechanics* (Oxford University Press, New York, 1987).
[5] T. S. van Erp, D. Moroni, and P. G. Bolhuis, J. Chem. Phys. **118**, 7762 (2003).
[6] M. S. Apaydin, D. L. Brutlag, C. Guestrin, D. Hsu, and J. C. Latombe, ''Stochastic roadmap simulation: An efficient representation and algorithm for analyzing molecular motion,'' Proceedings of RECOMB '02 (Washington, DC, 2002), pp. 12–21.
[7] G. Song, S. Thomas, K. Dill, J. M. Scholtz, and N. M. Amato, ''A path planning-based study of protein folding with a case study of hairpin formation in protein G and L,'' Proceedings of the Pacific Symposium on Biocomputing (Lihue, Hawaii, 2003).
[8] C. D. Snow, L. Qui, D. Du, F. Gai, S. J. Hagen, and V. S. Pande, Proc. Natl. Acad. Sci. U.S.A. **101**, 4077 (2004).
[9] C. Dellago, P. Bolhuis, and D. Chandler, J. Chem. Phys. **108**, 9236 (1998).
[10] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, J. Chem. Phys. **21**, 1087 (1953).
[11] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids* (Clarendon, Oxford, 1987).
[12] Y. Iba, Int. J. Mod. Phys. C **12**, 623 (2001).
[13] R. Du, V. Pande, A. Y. Grosberg, T. Tanaka, and E. Shakhnovich, J. Chem. Phys. **108**, 334 (1998).
[14] V. S. Pande, A. Y. Grosberg, T. Tanaka, and D. S. Rokhsar, Curr. Opin. Struct. Biol. **8**, 68 (1998).
[15] V. S. Pande and D. S. Rokhsar, Proc. Natl. Acad. Sci. U.S.A. **96**, 9062 (1999).
[16] L. Li and E. I. Shakhnovich, Proc. Natl. Acad. Sci. U.S.A. **98**, 13014 (2001).
[17] J. Gsponer and A. Caflisch, Proc. Natl. Acad. Sci. U.S.A. **99**, 6719 (2002).
[18] G. H. Golub and C. van Loan, *Matrix Computations*, 3rd ed. (The Johns Hopkins University Press, London, 1996).
[19] A. G. Cochran, N. J. Skelton, and M. A. Starovasnik, Proc. Natl. Acad. Sci. U.S.A. **98**, 5578 (2001).
[20] M. Shirts and V. Pande, Science **290**, 7220 (2000).
[21] A. R. Fersht, Proc. Natl. Acad. Sci. U.S.A. **99**, 14122 (2002).
[22] W. L. Jorgensen, D. S. Maxwell, and J. Tirado-Rives, J. Am. Chem. Soc. **118**, 11225 (1996).
[23] D. Qiu, P. S. Shenkin, F. P. Hollinger, and W. C. Still, J. Phys. Chem. A **101**, 3005 (1997).