An Overview and Practical Guide to Building Markov State Models

2

Gregory R. Bowman

The main purpose of this chapter is to provide a practical guide to building Markov models, with an emphasis on partitioning a molecule's conformational space into a valid set of states. This process is often referred to as constructing a state decomposition.

2.1 The Big Picture

The ideal state decomposition method would perform a truly kinetic clustering of a data set to accurately resolve the barriers between metastable states. Unfortunately, there is no simple means to calculate the average transition time between two arbitrary conformations.

One alternative would be to build states based purely on geometric criteria. However, this approach turns out to be inadequate because there is no physical reason that the locations of free energy barriers should correlate with geometric criteria. For example, two conformations with a 5 Å RMSD could fall within the same free energy basin if they only differ by pivoting of a hinge motion while another pair of conformations separated by the same distance could fall in different basins if they differ by strand pairings in a beta sheet. Mistakenly grouping together conformations that are not within the same free energy basin can create states with large internal free energy barriers. Such states will violate the Markov property because a system that enters the state on one side of the barrier will behave differently than one that enters on the other side, thereby introducing history dependence. Higher order Markov chains could be used to capture this history dependence, however, doing so greatly increases the number of parameters that must be determined, making it harder to obtain sufficient statistics. Moreover, people generally do not have nearly as well developed an intuition for processes with history dependence as we do for Markov models, so higher order models would provide less understanding.

At present, the most common approach for building MSMs is a two-stage process exploiting both geometry and kinetics [1–6]. In this two-stage approach, one uses a kinetically-relevant geometric clustering to create a starting point for a more purely kinetic clustering. By kinetically-relevant, I simply mean a clustering that only groups conformations together if the system can transition between them quickly relative to transitions between clusters.

The objective of the first stage is to create small volume elements in conformational space—called microstates—that are essentially the same structure using a geometric clustering. The motivation for starting with such a clustering follows that employed in the study of probability distribution functions, where one recognizes that the probability of a single point is vanishingly small and, therefore, works with small volume

G.R. Bowman (✉)
University of California, Berkeley 94720, USA
e-mail: gregoryrbowman@gmail.com

G.R. Bowman et al. (eds.), *An Introduction to Markov State Models and Their Application to Long Timescale Molecular Simulation*, Advances in Experimental Medicine and Biology 797,
DOI 10.1007/978-94-007-7606-7_2,

elements instead. At this stage, one would like to go out of their way to divide phase space as finely as possible to ensure that no microstate contains large free energy barriers. However, this objective is counterbalanced by the need to maintain sufficient statistics for each state such that transition probabilities between each pair of states can be estimated accurately. Given a set of microstates that meets these requirements, the transition probability between a pair of states can be calculated numerically by counting the number of times a simulation started in one of them and ended in the other because now two simulations have a finite probability of entering the same volume element. As discussed shortly, there are a number of ways to create and validate microstate models. Such models are excellent for making a quantitative connection with experiments because of their high resolution. However, they are often difficult to understand because they typically have tens of thousands of states.

To make a more understandable model, one can perform a kinetic clustering of a kinetically-relevant set of microstates to form larger aggregates—called macrostates—that correspond to free energy basins. One objective of this type of coarse-graining is to create mesoscale models that are still quantitative but are much more compact than the initial microstate model. These models may still be too complex to understand, however, the reduced state space makes them much easier to work with. A second objective is to coarse-grain the model so much that one can actually understand it. Often, these models will only be qualitatively correct—no longer able to make a quantitative connection with experiment. However, such extreme coarse-grainings are excellent for gaining an intuition for a system and generating new hypotheses to be tested with higher resolution models and, ultimately, with experiments.

To summarize, the key steps for building an MSM are

1. Choose an appropriate distance metric and cluster your simulation data into microstates.
2. Test the kinetic relevance of this clustering and choose an appropriate lag time (or observation interval) based on the Markov time of the model (smallest lag time that gives Markovian behavior).
3. Estimate the microstate model's transition probability matrix.
4. Coarse-grain the model to create either quantitative mesoscale models or qualitative models for guiding one's intuition.
5. Use the qualitative models to understand your system and the microstate or mesoscale models to model experiments.

Following is an explanation of the various alternatives for each of these steps. Throughout this discussion, a model refers to a transition probability matrix and one or more representative conformations from each state.

2.2 Clustering to Generate Microstates

The major objective of this step is to construct a kinetically-relevant clustering using geometric criteria. Such a clustering should only group together conformations the system can jump between rapidly. Many clusterings may satisfy this requirement, so there is not necessarily a single right answer. The resulting microstate model can then be used for making a quantitative connection with experiment or as a starting point for kinetic clustering.

Some of the key choices for this step are which distance metric to use, which clustering algorithm to use, how many clusters to generate, and which data to cluster.

2.2.1 Choosing a Distance Metric

It should come as no surprise that creating a kinetically-relevant clustering is best achieved with a kinetically-relevant distance metric. In particular, it is necessary for conformations separated by small distances to interconvert rapidly. Any distance metric that satisfies this requirement is sufficient given infinite data—i.e. the ability to create an infinitude of infinitely small states. However, one can typically make far better use of finite data by employing distance metrics that

best capture the relevant dynamics. For example, the opening angle may be a good choice for studying the opening and closing of a hinged-protein [7].

In leu of an obvious problem specific metric, the root-mean-square deviation (RMSD) between atoms is often a reasonable choice for protein dynamics [1, 2, 6]. Large RMSDs are hard to interpret, but two conformations separated by only a few Å are likely to interconvert rapidly. For extremely detailed models, the all-atom RMSD may be useful—though one must be careful about symmetry issues, like the invariance of Phe to a 180 degree flip. Basing the RMSD on α-carbons or all backbone heavy atoms is often sufficient though.

2.2.2 Choosing a Clustering Algorithm

Here, we briefly review a number of the clustering algorithms currently in use and their relative merits. There are many other options and there is great value in assessing their relative merits [5, 8]. For the purposes of this review, however, I hope only to describe a few of the most common options with the intent that this analysis will serve as a guide for evaluating other options.

2.2.2.1 *k*-Centers Clustering

In k-centers clustering, one tries to create a set of clusters with approximately equal radii by optimizing the objective function

$$\min_{\sigma} \max_{i} d\left(x_i, \sigma(x_i)\right) \tag{2.1}$$

where $\sigma(x)$ is a function that maps a conformation (x) to the nearest cluster center and $d(x, y)$ is the distance between two conformations x and y. The minimization occurs over all clusterings (σ) with k states and the max is taken over all conformations in the dataset. The radius of a cluster is just the maximum distance between any data point in that cluster and the cluster's center.

One advantage of k-centers is that it divides up conformational space more evenly than other algorithms by ensuring that states have similar radii [9, 10]. Intuitively, one can think of this algorithm as creating clusters with approximately equal volumes. However, one must take care not to take this too literally as very small variations in the radii of clusters in high-dimensional spaces can give rise to huge variations in their volumes. Having a more or less even division of conformational space into microstates is of value because it helps avoid situations where some regions are unnecessarily divided into an excess of states while other regions are not sufficiently broken-up to avoid large internal free energy barriers. Properties of the model, like the slowest relaxation time, should also be insensitive to the exact clustering as long as one purposefully over-divides conformational space into a large number of states.

Another advantage of k-centers is that there is an extremely fast, deterministic approximation to this algorithm [9, 10]. This approximate algorithm has an $O(kN)$ runtime, where k is the number of clusters and N is the number of conformations being sampled, so it is applicable to even extremely large data sets. This algorithm works as follows (Fig. 2.1):

1. Choose an arbitrary point as the initial cluster center and assume all other data points are initially in that cluster (Fig. 2.1A).
2. Calculate the distance from every other data point to the current cluster center.
3. Select the furthest point from any existing cluster center as the next cluster center (Fig. 2.1B).
4. Calculate the distance from every data point to this new cluster center and reassign any of them that are closer to the new center than their previous cluster center to the new cluster (Fig. 2.1B).
5. Repeat steps 3 and 4 until some cutoff criterion—like the number of clusters or maximum size of any cluster—is reached (Fig. 2.1C).

The most appropriate cutoff criterion depends on the process of interest. For example, creating clusters until each state has a radius of less than 3 Å RMSD is often an appropriate starting point for protein folding, where the relevant conformational space is huge and a rather coarse partitioning will do. For more subtle conformational changes where there is a small space and more

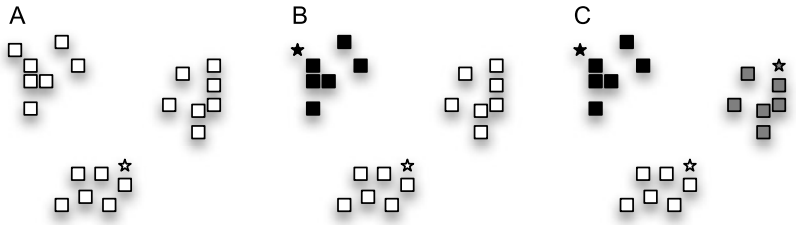

Fig. 2.1 An example of k-centers clustering of a set of data points (*squares*). (**A**) First, a random data point (*white star*) is chosen as the initial cluster center and all data points are assigned to it (*white squares*). (**B**) Next, the data point furthest from the previous cluster center is chosen as the next cluster center (*black star*). All the data points that are closer to the new cluster center than any existing center are assigned to the new center (*black squares*). (**C**) The algorithm continues choosing the data point that is furthest from any existing center (in this case the *gray star*) and assigning data points that are closer to it to the new center (*gray squares*)

detail is required, a cutoff of 1 Å may be a better starting place.

One disadvantage of k-centers is that building a viable set of microstates with this algorithm often requires creating a large number of clusters. For example, modeling the folding of a 35 residue variant of the villin headpiece—one of the smallest known proteins—still required 10,000 states [11]. As a result, one needs a great deal of sampling to ensure adequate statistics for each state. The large number of states also leads to large microstate transition matrices that can be computationally demanding to work with. A more ideal algorithm would use kinetic criteria to have larger or smaller states as needed to accurately capture the underlying landscape. Finally, the "centers" created by k-centers are not necessarily anywhere near the geometric center of the cluster. Instead, they are often on the periphery of the cluster because the approximate algorithm presented here is always choosing the data point furthest from all the existing cluster centers as the next one and, therefore, is biased towards choosing data points at the very edge of the space sampled (Fig. 2.1). Thus, the cluster centers are not necessarily representative of the data assigned to them. One must use other strategies to identify representative conformations for a cluster created with k-centers, like drawing a few random conformations from it.

2.2.2.2 k-Medoids Clustering

The k- medoids algorithm minimizes the average distance between data points and the center they are assigned to by optimizing

$$\frac{1}{N}\sum_i d\left(x_i, \sigma(x_i)\right)^2 \tag{2.2}$$

where N is the number of data points, $\sigma(x)$ is a function that maps a conformation (x) to the nearest cluster center, and $d(x, y)$ is the distance between two conformations x and y.

The k-medoids algorithm is very similar to k-means but with the important difference that only data points can be cluster centers. In k-means, the cluster center is the average of all the data points belonging to that cluster. However, taking the average of a number of protein conformations does not make physical sense as it can easily lead to unphysical behavior like steric clashes and extremely unlikely bond lengths/angles. Thus, k-medoids is preferable.

One advantage of k-medoids over k-centers is that k-medoids tends to create clusters with a more equal number of samples. For example, if a data set has a densely sampled region and a sparsely sampled region, then k-medoids will tend to place more clusters in the densely sampled region. This feature is useful in that it helps avoid states with too few counts to make statistically reliable estimates of the transition probabilities to other states. However, k-medoids may also over-divide some regions of conformational space and under-divide others. For in-

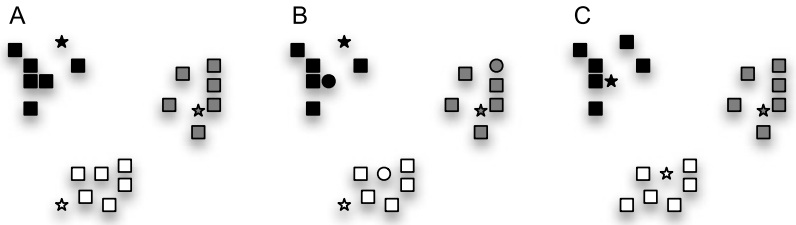

Fig. 2.2 An example of k-medoids clustering of a set of data points (*squares*). (**A**) First, the user decides how many clusters to construct (in this case $k = 3$). Then k random data points are chosen as initial cluster centers (the *three stars*). All data points are then assigned to the closest center (as indicated by the color coding). (**B**) Next, a random data point in each cluster is proposed as a new cluster center (*circles*). (**C**) If the newly proposed center is closer on average to all the data points in the cluster than the previous center, then it is chosen as the new center for that cluster (as in the *black* and *white clusters*). Otherwise, the newly proposed center is rejected and the previous center is kept (as in the *gray cluster*). Finally, all the data points are reassigned to their closest center. Note that this is an extremely contrived example. Usually there will not be such a clear distinction between clusters, the number of clusters will be unclear, and the initial cluster centers may end up being very close to one another, requiring a number of iterations of updating before convergence to a reasonable set of centers

stance, in protein folding, k-medoids is likely to create many clusters in the folded ensemble and very few clusters in more sparsely populated unfolded regions. Therefore, one will get essentially redundant clusters in the folded ensemble while mistakenly grouping together kinetically distinct unfolded conformations into states that violate the Markov assumption. However, this problem may not arise for other processes, like conformational changes, where the relevant regions of conformational space may be more evenly sampled.

The k-medoids algorithm works as follows (Fig. 2.2):

1. Randomly choose k conformations as the initial cluster centers (Fig. 2.2A).
2. Assign each data point to the closest center.
3. For each cluster C, propose a random data point $z \in C$ as the new center (Fig. 2.2B) and evaluate the change using

$$\sum_{x_i \in C} d(x_i, z)^2 \tag{2.3}$$

   If the newly proposed center reduces the objective function compared to the previous center, then replace the current cluster center with z (Fig. 2.2C).
4. Repeat steps 2 and 3 for a specified number of iterations or until the algorithm converges to a stable result.

To speedup the algorithm further, it is common to propose a number of possible new centers for each cluster during step 2.

One advantage of k-medoids is that the resulting centers are actually representative of the data assigned to them because they lie at the center of the cluster. A disadvantage is that the number of clusters must be chosen *a priori*, compared to k-centers where it is possible to choose a physically meaningful criterion for determining the number of states.

2.2.2.3 Hybrid k-Centers/k-Medoids Clustering

A hybrid approach has been developed to strike a balance between the strengths and weaknesses of the k-centers and k-medoids algorithms [4]. This algorithm simultaneously optimizes the objective functions for both k-centers (Eq. (2.1)) and k-medoids (Eq. (2.2)) as follows:

1. Perform an approximate k-centers clustering, as in Sect. 2.2.2.1.
2. Update the centers with a number of iterations of the k-medoids update step (Steps 2 and 3 of the k-medoids algorithm in Sect. 2.2.2.2), rejecting any proposed moves that increase the k-centers objective function in Eq. (2.1).

This hybrid approach appears to be a good, general purpose method for building microstate models. For example, like k-centers, this method

still gives a more even discretization of conformational space than a pure k-medoids clustering and one can specify a physically meaningful criterion for determining the number of states to create. The k-medoids update step also results in cluster centers that are representative of the data assigned to them. More importantly, using this update step shifts the centers to the densest regions of conformational space within a state, leading to better resolution of the boundaries between states. As a result, this algorithm yields shorter Markov times with fewer states. Having fewer states means each one has better statistics and less data is required to parameterize the model.

There is still room for improving upon this hybrid approach though. For instance, this algorithm still tries to avoid states with large internal barriers by creating a large number of clusters. As discussed previously, parameterizing models with more states requires more data to obtain sufficient statistics and large transition matrices can be challenging to work with.

2.2.3 Subsampling

One final question that deserves some consideration is which data to cluster. In an ideal case, where one could perform a purely kinetic clustering of simulation data, the answer to this question would be simple: cluster all the data. However, using all the data is not always optimal when starting off with a geometric clustering. For example, an RMSD-based k-centers clustering will select every structural outlier as a cluster center before starting to subdivide the well-sampled regions of conformational space. There are also practical limitations, like the number of conformations that can be stored in memory on typical computers.

Using a subsample of ones data to define a set of microstates can lead to better models because this strategy reduces the impact of outliers [11]. Put another way, clustering a subsample of one's data focuses the cluster centers on the better sampled regions of conformational space. After defining a state space based on a subsample of the data, one can then assign all the data to these microstates, thereby obtaining more statistics. Outliers will then be absorbed into the closest cluster, where they will have little impact on the quality of the model. For protein folding—which typically occurs on time scales of a microsecond or longer—a fruitful procedure has been to store conformations every 1 ns, cluster conformations sampled at a 10 ns interval, and then assign all the data to the resulting microstates.

2.3 Estimating Transition Matrices

In theory, estimating a transition matrix should just be a matter of counting. The first step is to assign data to clusters, which we will number from 0 to $n - 1$. Now each trajectory can be thought of as a series of microstate assignments rather than as a series of conformations. The number of transitions between each pair of states can then be counted and stored as a transition count matrix (C), where C_{ij} is the number of transitions observed from state i to state j. With infinite data, one could just use the maximum likelihood estimate for the transition probability between each pair of states to convert the transition count matrix into a transition probability matrix (T). That is,

$$T_{ij}(\tau) = \frac{C_{ij}}{\sum_k C_{ik}} \tag{2.4}$$

where τ is the lag time of the model. However, in practice, estimating transition matrices is complicated by a number of issues, like finite sampling and imperfections in microstate definitions.

2.3.1 Counting Transitions

Counting transitions sounds like a simple task, but there are actually a variety of options that must be considered. First of all, one must choose a lag time at which the model satisfies the Markov assumption—as discussed in the next section. Choosing an appropriate lag time actually requires estimating transition matrices at a variety

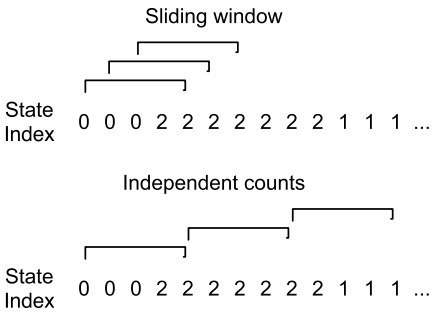

Fig. 2.3 An example of the two methods of counting transitions, assuming a 4-step lag time. Each *panel* shows a trajectory as a series of state indices (i.e. assuming the states have been numbered from 0 to $n-1$). The *top panel* shows how the first three transitions would be counted using the sliding window approach (*brackets*). All three transitions are from state 0 to 2. The *bottom panel* shows how to ensure independent counts. The three transitions (indicated with *brackets*) are from state 0 to 2, from state 2 to 2, and from state 2 to 1

of lag times, so we will cover the process of estimating these matrices first.

In an ideal case, where one has an excess of data relative to the slowest relaxation time in the system, one could simply look at independent transitions at the lag time (τ). That is, one could look at the state indices at an interval of τ. As shown in Fig. 2.3, one could then count transitions as $\sigma(0) \rightarrow \sigma(\tau)$, $\sigma(\tau) \rightarrow \sigma(2\tau)$, $\sigma(2\tau) \rightarrow \sigma(3\tau)\ldots$ where $\sigma(t)$ is the state index of the simulation at time t. However, with finite data, this can lead to imprecise estimates of transition probabilities due to model uncertainty.

Practically, it is often useful to use a sliding window approach. In this approach, one assumes conformations were sampled at a regular interval Δ, where $\Delta < \tau$. For example, one could store conformations every 100 ps and have a lag time of 10 ns. As shown in Fig. 2.3, one could then count transitions as $\sigma(0) \rightarrow \sigma(\tau)$, $\sigma(\Delta) \rightarrow \sigma(\Delta+\tau)$, $\sigma(2\Delta) \rightarrow \sigma(2\Delta+\tau)\ldots$ where $\sigma(t)$ is the state index of the simulation at time t. The sliding window approach will give a more precise estimate of transition probabilities but will lead to underestimates of model uncertainty (see Sects. 4.1 and 5.1).

The sliding window approach is recommended for estimating maximum likelihood transition matrices as precisely as possible. Counting independent transitions should be used when estimating model uncertainty.

2.3.2 Detailed Balance

Another major issue is satisfying detailed balance—also called microscopic reversibility. That is, every time there is a transition from state i to j, there should also be a compensating transition from state j to i. Without this property, one would get source and sink states that would prevent the model from accurately describing long time scale behavior.

Poorly sampled microstates are one issue that can break detailed balance. In particular, some states may have a single transition into or out of them. A simple maximum likelihood estimate of transition probabilities would then turn these states into sources or sinks. Therefore, it is often useful to trim off these states [4, 12, 13].

One must also satisfy detailed balance between every pair of states. One simple way of enforcing detailed balance is to assume that every time there is a transition from state i to j, there must be a corresponding transition from state j to i. The maximum likelihood estimate of the number of transitions from state i to j is then

$$\hat{C}_{ij}(\tau) = \frac{C_{ij} + C_{ji}}{2} \tag{2.5}$$

where $\hat{C}_{ij}$ is an estimate of the reversible counts from state i to j and C_{ij} are the number of transitions actually observed. This method is perfectly valid if one has true equilibrium sampling. Furthermore, it is extremely robust in the sense that this algorithm will always give an estimate of the transition probability matrix that is consistent with the original data. However, if one has limited data, as is often the case, then one's estimate of the transition probability matrix will be extremely biased towards the starting conditions of their simulations. For example, the equilibrium probability of each state will just be proportional to the number of data points in it.

One alternative is to use maximum likelihood methods that try to estimate the reversible transition probability matrix that is most likely to

have given rise to the observed data [4, 5, 11]. These methods will be described in more detail in Sect. 4.6. Here, I will just note that these methods are extremely powerful when they work. However, at present, they often fail to converge or converge on transition probability matrices that over-emphasize the importance of poorly sampled states.

2.3.3 Ergodicity

One final point is that a valid MSM must be ergodic. That is, the network of states must be fully connected. Physically, this means that it is possible to reach any state from an arbitrarily chosen starting state. Disconnected components can arise when different initial conformations are employed and sufficient sampling is not obtained to observe mixing (or overlap) between simulations started from different structures. When this happens, it is impossible to determine the relative equilibrium probabilities of disconnected components or the probabilities of transitions between them. Two possible solutions are (1) to discard all but one of the components (typically the largest one) [4, 12, 13] or (2) to collect more data until the network of states becomes completely connected.

2.4 Model Validation and Lag Time Selection

Before drawing any conclusions from a model, it is crucial to test whether or not it is kinetically-relevant and to choose an appropriate lag time. The dynamics of a perfectly specified system, including solvent degrees of freedom and every atom's velocity, is certainly Markovian because the next conformation is simply a deterministic function of the system's current state. However, even microstate models effectively coarse-grain the system. For example, conformations are grouped together and water degrees of freedom are often ignored. Therefore, both microstate models and coarse-grainings thereof may only be Markovian at longer time scales, if at all. As discussed previously, large internal barriers can lead to models that violate the Markov assumption.

2.4.1 Tests Based on the Chapman-Kolmogorov Equation

Many tests of model validity make use of the Chapman-Kolmogorov equation

$$T(n\tau) = T(\tau)^n \tag{2.6}$$

where n is an integer number of steps, each one lag time τ in length. This equation captures the fact that taking n steps with an MSM with a lag time of τ should be equivalent to an MSM with a lag time of $n\tau$.

Plotting the relaxation time scales of a model—also called its implied time scales—as a function of the lag time is one use of the Chapman-Kolmogorov equation that provides some model validation and a means of choosing an appropriate lag time [14]. As will be discussed in more detail in Sect. 3.2, the relaxation times of a model are a function of the eigenvalues of its transition probability matrix

$$t_i = -\frac{\tau}{\ln \lambda_i} \tag{2.7}$$

where t_i is a relaxation time, τ is the lag time, and λ_i is an eigenvalue. Based on the Chapman-Kolmogorov equation, the relaxation times for a Markov model with a lag time of $n\tau$ should be the same as those for a Markov model with a lag time of τ

$$\begin{aligned} t_i &= -\frac{n\tau}{\ln \lambda_{i,T(n\tau)}} = -\frac{n\tau}{\ln \lambda_{i,T(\tau)}^n} \\ &= -\frac{n\tau}{n \ln \lambda_{i,T(\tau)}} = -\frac{\tau}{\ln \lambda_{i,T(\tau)}} \end{aligned} \tag{2.8}$$

where $\lambda_{i,T(\tau)}$ is an eigenvalue of $T(\tau)$. Therefore, examining a plot of the relaxation timescales as a function of the lag time should give an indication of when a model starts to satisfy the Markov assumption, if at all. Beyond the Markov time (the smallest lag time that gives Markovian behavior), the relaxation time scales should be level,

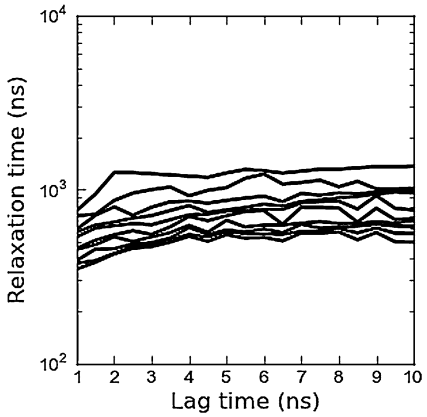

Fig. 2.4 An example relaxation timescale (or implied timescale) plot with a Markov time of ~2 ns. Data comes from Ref. [15]

as shown in Fig. 2.4. If the relaxation time scales never level-off, then it is likely that one or more states have large internal barriers and a new state decomposition is necessary, and possibly more data as well. Unfortunately, this is a rather subjective test, particularly when dealing with finite statistics.

One can also go beyond plots of the relaxation timescales and test the Chapman-Kolmogorov equation on a state by state basis [5]. More details on this approach are given in Sect. 4.8.

2.4.2 Correlation Function Tests

Comparing correlation functions from the raw data and an MSM is one alternative to Chapman-Kolmogorov-based tests when one has sufficiently long simulations. To calculate correlation functions for a single long trajectory, one first calculates some property of interest for each snapshot—like the RMSD to a crystal structure—and then calculates

$$c(t) = \langle \theta(0)\theta(t) \rangle \tag{2.9}$$

where $\theta(t)$ is the observable at time t.

One can also calculate a (normalized) correlation function for an MSM using

$$c(n) = \frac{\sum_{i=1}^{N} \lambda_i^n (\boldsymbol{\theta} \cdot \boldsymbol{\phi}_i)^2}{\sum_{i=1}^{N} (\boldsymbol{\theta} \cdot \boldsymbol{\phi}_i)^2} \tag{2.10}$$

where n is the number of steps (t/τ), N is the number of states, $\boldsymbol{\theta}$ is a vector of observables for each state, and $\boldsymbol{\phi}_i$ is the ith left eigenvector [16]. Unfortunately, this test cannot be used if you only have short simulations. One needs at least one long simulation compared to the relaxation time of the model to calculate the reference correlation function.

2.5 Coarse-Graining to Generate Macrostates

As discussed in Sect. 2.1, there are a number of advantages to coarse-graining microstate models by merging rapidly mixing microstates into larger macrostates. First of all, one can sometimes build mesoscale models that are just as quantitatively predictive as the original microstate model but are far more compact. Secondly, one can build models with few enough states that they are comprehendible and can be used to gain an intuition for a system and generate hypotheses, though they may no longer be quantitatively predictive.

Two major questions have to be addressed to build these coarse-grained models. First, how should one determine which microstates to merge together? Secondly, how many macrostates should one build?

Here, we review a number of methods that have been developed to answer these questions.

2.5.1 PCCA

Perron Cluster Cluster Analysis (PCCA) uses the eigenspectrum of a transition probability matrix to construct coarse-grained models [17, 18]. This method derives its name from the Perron-Frobenius theorem, which states that a real square matrix with positive entries (e.g. a transition probability matrix) has a unique largest real eigenvalue and that the corresponding eigenvector has strictly positive components. The term Perron Cluster refers to a set of eigenvalues clustered near the largest eigenvalue and separated from the rest of the eigenspectrum by a reasonable gap. As discussed shortly, the eigenvectors

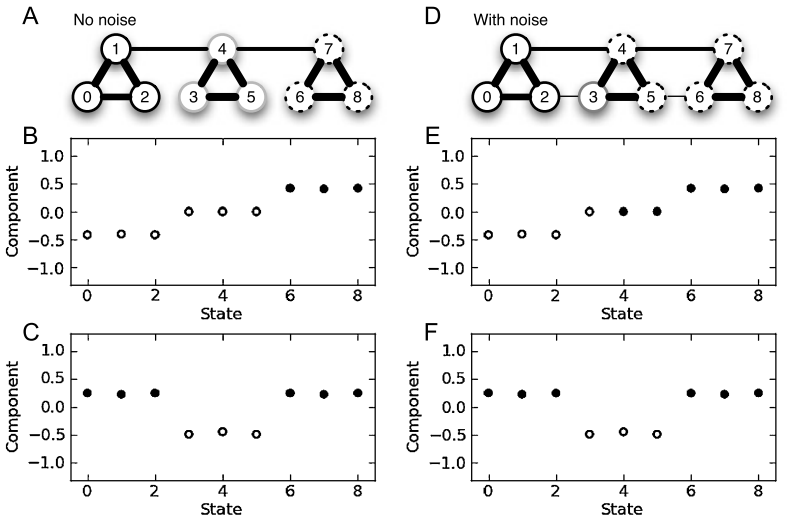

Fig. 2.5 Two simple models demonstrating the power and pitfalls of PCCA. (**A**) A simple model with well-sampled transitions. Each of the nine microstates has 1,000 self-transitions. Each *thick line* corresponds to 100 transitions and the *medium weight lines* correspond to 10 transitions. This model can be coarse-grained into three macrostates consisting of microstates 0–2 (*black outline*), 3–5 (*gray outline*), and 6–8 (*dotted outline*). Panels (**B**) and (**C**) show the second and third eigenvectors of this simple model, respectively. *Open circles* are used for eigenvector components that are less than or equal to zero and filled circles are used for components that are greater than zero. (**D**) The same simple model with two poorly sampled transitions (noise) added between states 2–3 and states 5–6. These transitions have only a single count. Their presence barely changes the model's eigenvectors. However, they alter the sign structure of the second eigenvector (panel (**E**)) and, therefore, PCCA finds the wrong coarse-graining into three macrostates: microstates 0–2 (*black outline*), 3 (*gray outline*) and 4–8 (*dotted outline*). Panels (**B**) and (**C**) show the second and third eigenvectors of the simple model from panel (**D**), respectively. If you try this model on your own computer, note that your results may vary slightly due to the symmetry of the system

corresponding to the Perron Cluster can be used to coarse-grain an MSM.

We begin with a discussion of PCCA because it highlights many of the issues that must be considered when coarse-graining MSMs. PCCA was also one of the first methods for coarse-graining MSMs and, therefore, provided at least some of the intellectual inspiration for many of the other methods. However, the other methods presented here are likely to perform better than PCCA, which does not provide a numerically robust clustering.

As discussed previously, the eigenvalues of a transition probability matrix can be converted into time scales. The corresponding eigenvectors describe what transitions are occurring on each of these time scales [17]. The largest eigenvalue (λ_1) is always 1 for a model that is connected and obeys detailed balance. The components of the corresponding eigenvector are proportional to the equilibrium populations of each state. The remaining eigenvalues ($\lambda_n < \lambda_{n-1} < \cdots < \lambda_2 < 1$) are real-valued and can be converted into time scales using Eq. (2.7). The corresponding right eigenvector describes what is happening on this time scale. That is, states with negative eigenvector components are interconverting with states with positive components and the magnitude of these components is proportional to the state's degree of participation. The left eigenvectors contain the same information but weighted by the equilibrium population of each state.

In PCCA, one starts off with all microstates merged into a single macrostate and then iteratively breaks the most kinetically diverse macrostate into two smaller states based on the next slowest right eigenvector [17, 18]. As an example, let's consider the model shown in Fig. 2.5A. By eye, it is clear this model can be divided into 3 macrostates: one containing mi-

crostates 0–2, one containing microstates 3–5, and one containing microstates 6–8. To find these states with PCCA, one would begin by using the second eigenvector (ψ_2, Fig. 2.5B) to split the microstates into one group with components less than or equal to zero (open circles in Fig. 2.5B) and one group with components greater than zero (filled circles in Fig. 2.5B). This splitting would give rise to two macrostates, one containing microstates 0–5 and another containing microstates 6–8. Next, PCCA chooses the group with the greatest spread in eigenvector components and uses the next slowest eigenvector (ψ_3) to divide this group in two. In this example, PCCA would select the group containing microstates 0–5 because it has states with components of ψ_2 ranging from about -0.4 to 0, whereas the other group only has states with components of about 0.4. Using ψ_3, PCCA would then split this group into two smaller groups containing microstates 0–2 and 3–5, respectively. This second split gives us the natural grouping into three macrostates we can see by eye, demonstrating the utility of this automated procedure. Further iterations of this algorithm could then be used to create more macrostates.

One attractive feature of this algorithm is that it provides a natural way to choose how many states to construct. If the system of interest has a well defined set of free energy basins—i.e. with large barriers between them and significantly smaller barriers within them—then the system will exhibit a separation of time scales. That is, there should be a Perron Cluster of eigenvalues near 1 that are separated from the rest of the eigenvalue spectrum by a reasonable gap. For example, the eigenvalues of our simple model from Fig. 2.5A are 1.0, 0.997, 0.992, 0.752, 0.750, 0.750, 0.750, 0.746, 0.735. There is a clear gap between the third and fourth eigenvalues of this model, and this gap would become even clearer after converting the eigenvalues into time scales. This gap indicates a separation of time scales that permits a three state macrostate model to capture the slowest relaxation time scales of the system. In general, if there is a gap after the nth eigenvalue (counting the eigenvalue of 1), then one should be able to construct a reasonable macrostate model with n states. Unfortunately, many real world systems do not have a clear separation of time scales. Instead, they have a continuum of eigenvalues. In such cases, the number of macrostates is best seen as an adjustable parameter one can vary depending on the properties of the model they are interested in.

One major limitation of PCCA is that it can suffer from propagation of error when not all microstates participate strongly in each eigenmode [19]. For example, in the simple model from Fig. 2.5A, microstates 3–5 have zero components in the second eigenvector, indicating they do not participate in the slowest relaxation process. This simple model is purposefully very clean, so all three of these microstates were placed in the same group during the first splitting PCCA performed based on the second eigenvector. However, in real world scenarios, one rarely gets states with eigenvector components of exactly zero because of factors like poorly sampled transitions. States that do not participate strongly in a given eigenmode (i.e. have eigenvector components that are nearly zero) will be assigned to macrostates rather arbitrarily, leading to compounding error as more eigenvectors are considered. For example, the simple model in Fig. 2.5D is the same as the one we've used so far but with the addition of two noisy transitions with only a single count between states 2–3 and states 5–6. In the second eigenvector of this model, microstates 3–5 have very small magnitude eigenvector components with different signs (Fig. 2.5E). Therefore, splitting this model into two groups based on the second eigenvector gives one set containing microstates 0–3 and another containing microstates 4–8. Despite the fact that microstates 3–5 should form a single macrostate, they have already been split apart at this early stage. As PCCA considers more eigenvectors, it will propagate this error. In more complicated models, new errors can also be introduced at each stage due to weakly participating states. Unfortunately, there is often a continuum of eigenvector components, so there is currently no clear way to separate weakly participating states from strongly participating ones and deal with the weakly participating ones separately.

A number of heuristic methods have been introduced to fix this problem but none are entirely satisfactory [1]. For example, intuitively, a partitioning into metastable states should maximize the self-transition probability of each state. Doing so is equivalent to minimizing the transition rates between states, or placing the boundaries between states along the largest free energy barriers. Therefore, one can try to correct for the errors introduced during PCCA by doing a simulated annealing procedure to maximize the total metastability (Q) of the model

$$Q = \sum_{n} T_{ii} \tag{2.11}$$

where n is the number of macrostates and T_{ii} is the self-transition probability for state i. In such a procedure, one tries randomly assigning microstates to new macrostates and each proposed move is accepted or rejected according to a Monte Carlo criterion. That is, moves that increase the metastability are always accepted and moves that reduce it are only accepted with some small probability. This procedure often works for simple models but becomes intractable for real world models because it can converge on non-sensical results or even completely fail to converge to a stable solution.

PCCA is also prone to handle poorly sampled states and transitions improperly because it does not account for statistical uncertainty in a model [21]. For example, it is common for the clustering algorithms described in Sect. 2.2 to make conformations that are geometric outliers into their own microstates [11]. These states will have very few transitions to other states and, therefore, will appear to be separated from them by large free energy barriers. As a result, PCCA will often make these poorly sampled microstates into singleton macrostates—i.e. macrostates containing a single microstate. Human examination of these states, however, often reveals that they are unlikely to be physically meaningful.

2.5.2 PCCA+

PCCA+ is a more robust version of PCCA that avoids the pitfall of propagation of error [19, 20]. This improvement is accomplished by considering the relevant eigenvectors simultaneously instead of sequentially. More specifically, PCCA+ tries to find a set of indicator functions that best reproduces the n slowest dynamical eigenvectors. For example, to construct a three-state macrostate model for the simple model in Fig. 2.5D, PCCA+ would consider the second and third eigenvectors simultaneously. PCCA+ would then fit these eigenvectors with three step functions: one that is 1 in states 0–2 and 0 elsewhere, a second that is 1 in states 3–5 and 0 elsewhere, and a third that is 1 in states 6–8 and 0 elsewhere. The details of how this optimization is achieved are very similar to spectral clustering and are described in Ref. [19].

While PCCA+ does not suffer from the propagation of error that occurs in PCCA, this method still relies on a maximum likelihood estimate of the transition probability matrix. Therefore, PCCA+ still tends to create singleton macrostates. Furthermore, PCCA+ can require quite a bit of memory, so creating mesoscale models is often computationally intractable.

2.5.3 SHC

Super-level-set hierarchical clustering (SHC) tries to deal with model uncertainty by treating low population states differently from high population ones [22]. Inspired by developments in topological data analysis, SHC first divides all the microstates into sets with similar populations (called level-sets) [23]. PCCA or PCCA+ is then used to divide each set into macrostates. Finally, overlap between the macrostates at each level is used to stitch these models together. Typically, PCCA(+) is not applied to the least populated states. Instead, these are just lumped into the macrostate they transition to most quickly, thereby avoiding creating singleton macrostates.

One added benefit of this approach is that the hierarchy of models SHC creates can give insight into the hierarchy of free energy basins that actually exist in the underlying free energy landscape.

For example, macrostates from the most populated level correspond to the deepest free energy basins. Some of the macrostates at less populated levels simply correspond to the same macrostates that are present at more populated levels. However, some reflect less populated intermediates between these deeper minima and can provide insight into how the system transitions between the most populated free energy basins.

SHC could benefit greatly from a more formal justification. One philosophical short coming is that SHC was inspired by methods that make use of density level sets (sets of data with approximately equal densities). However, as discussed earlier, estimating densities in high-dimensional spaces is extremely difficult. A more formal justification for breaking the microstates into levels would be preferable. One practical implication of this short-coming is that there is no clear way to define the level sets *a priori*. Instead, one must make a rather arbitrary choice and then try varying the density level sets to check for robustness. Therefore, SHC requires more computation than a single application of PCCA or PCCA+. A more formal justification for this method could provide insight into how to choose the density level sets and make this an extremely powerful method though. New work on framing SHC in terms of a Nystrom expansion of the transition probability matrix may provide such a formal justification.

2.5.4 BACE

More recently, a Bayesian agglomerative clustering engine (BACE) has been developed for dealing with uncertainty in a more automated fashion [21]. BACE exploits the observation that rapidly mixing states should also be kinetically similar—that is, they should have similar transition probabilities—to determine which states to lump together. The algorithm works by iteratively merging the most kinetically similar states, as judged by a Bayes factor for determining how likely the transitions observed for each state are to have come from the same underlying distribution

$$\ln \frac{P(\text{different}|C)}{P(\text{same}|C)} \approx \hat{C}_i \mathscr{D}(p_i \| q) + \hat{C}_j \mathscr{D}(p_j \| q) \tag{2.12}$$

where $P(\text{different}|C)$ is the probability the counts (C) from states i and j came from different underlying probability distributions and $P(\text{same}|C)$ is the probability they came from the same distribution, $\hat{C}_i$ is the number of counts originating in state i, $\mathscr{D}(p_i \| q) = \sum_k p_{ik} \ln \frac{p_{ik}}{q_k}$ is the relative entropy between probability distribution p_i and q, p_i is a vector of maximum likelihood transition probabilities from state i, and $q = \frac{\hat{C}_i p_i + \hat{C}_j p_j}{\hat{C}_i + \hat{C}_j}$ is the vector of expected transition probabilities from combining states i and j. Deriving this expression involves integrating over all possible transition probability distributions out of each state, so the method naturally takes into account uncertainty in the microstate model's transition probability matrix.

In addition to outperforming many other methods, BACE has an appealing information theoretic interpretation and provides a way to determine which levels of the hierarchy of models it creates are most deserving of further analysis. Specifically, Eq. (2.12) is identical to the Jensen-Shannon divergence, a popular measure from information theory [24]. Therefore, BACE can be interpreted as creating the coarse-graining that retains the maximum information about the original model's kinetics. The BACE Bayes factor also provides a means to determine how many states to create. One can simply monitor the Bayes factor as one merges states and watch for dramatic jumps. Models preceding these jumps are particularly deserving of further analysis because further coarse-graining greatly reduces the model quality.

Fully characterizing the strengths and weaknesses of BACE compared to other methods will require further application of this method. BACE is probably extremely useful for building mesoscale models because it can build them quickly and accurately. However, it may be less useful for building extremely coarse-grained

models. The fewer the states one requests from BACE, the more iterations it must run and the longer the algorithm takes to complete. BACE could also suffer from propagation of error, as seen in PCCA, as a mistake early on will never be corrected later.

2.6 Recommended Protocol

At present, one of the most robust and widely used—but not necessarily optimal!—protocols for modeling proteins and other biomolecules is:

1. Cluster a simulation data set with the hybrid k-centers/k-medoids algorithm based on the RMSD between backbone heavy atoms, ensuring that every cluster has a radius on the order of a few Å.
2. Validate the kinetic relevance of this clustering and choose an appropriate lag time based on the model's relaxation time scales (or implied time scales) as a function of the lag time. For each lag time, estimate the transition matrices by:
   a. Removing states that only have one transition with any other state.
   b. Counting transitions using a sliding window and assuming that for every transition from state i to j, there is a corresponding transition from j to i.
3. Use the transition probability matrix at the desired lag time and representative conformations from each state to model experiments.
4. Coarse-grain the model with PCCA+ to gain an intuition for the system. Make sure to test how quantitative the coarse-grained model is by examining how closely the macrostate model's relaxation times agree with the microstate model.

2.7 Advanced Topics and Future Directions

Before moving on to the next chapter, I would like to briefly review a number of advanced topics and list some of the future directions that could lead to more effective methods for building MSMs.

2.7.1 Seeding

In seeding, one uses an inexpensive method to choose a variety of starting conformations for simulations that will later be used to build an MSM [25]. Intuitively, seeding allows one to explore a wider swath of conformational space more quickly than would be possible by starting every simulation from a single conformation, like a crystal structure. Ideally, one would like to find the greatest possible variety of relevant conformations. One way of doing this is to use generalized ensemble simulations like replica exchange to quickly explore conformational space and then use random conformations from these simulations as starting points for constant temperature simulations. This procedure helps focus the starting conformations on thermodynamically relevant regions of phase space. Less thermodynamically relevant conformations should quickly relax to more populated regions of conformational space.

When using seeding, one must be careful to ensure that the resulting model is connected. Simulations started from conformations that are too kinetically distant from any of the other starting points may never overlap with the rest of the trajectories, making it impossible to determine transition rates between them or their relative equilibrium populations.

2.7.2 Cores

Fluctuations at the tops of barriers between states can lead to recrossing events where the system appears to rapidly jump back and forth between the start and end state [26]. These recrossing events can lead to over-estimates of the transition rates between states if they are not dealt with properly.

Cores are one way of reducing the affect of recrossing [16, 27]. The basic idea is to define a core region within each state, leaving a no-man's land between each pair of states. Transitions are only counted when a trajectory leaves the core of one state and enters the core of another. A trajectory that makes an excursion into no-man's land

but then returns to the core it started in before entering the core of any other state is said never to have left its initial state.

2.7.3 Comparing Multiple Sequences

In addition to providing insight into a single system, MSMs are also a powerful way of comparing different systems. For instance, in protein folding, there is great interest in comparing slight variations of a single protein and how the mutations that distinguish them change properties like the folding rate or stability of the native state. For this to be possible, it is essential that a common state space be used. For example, one can construct a common microstate space by clustering two protein sequences using a set of atoms that they share in common [28]. Properties like the equilibrium populations of a set of states or the transition rates between them can then be compared between the two systems.

2.7.4 Open Challenges

MSM methods are sufficiently well developed to pursue many exciting applications. However, there is still a great deal of room for further methodological improvements. Here, I list just a few of them.

1. As discussed previously, one would ideally like to build MSMs using a truly kinetic distance metric from the beginning. New clustering methods or distance metrics that better reflect a system's kinetics would be of tremendous value.
2. Many of the methods for validating MSMs and choosing important parameters, like the lag time, are very subjective. Quantitative approaches to model validation would allow for more automatic model construction.
3. More robust methods for estimating transition matrices that satisfy detailed balance would also be useful. Current methods are either too biased or too unreliable.
4. There is still a need for more efficient and accurate coarse-graining methods.

References

1. Chodera JD et al (2007) Automatic discovery of metastable states for the construction of Markov models of macromolecular conformational dynamics. J Chem Phys 126:155101
2. Bowman GR, Huang X, Pande VS (2009) Using generalized ensemble simulations and Markov state models to identify conformational states. Methods 49:97–201
3. Noé F et al (2009) Constructing the equilibrium ensemble of folding pathways from short off-equilibrium simulations. Proc Natl Acad Sci USA 106:19011–19016
4. Beauchamp KA et al (2011) MSMBuilder2: modeling conformational dynamics on the picosecond to millisecond scale. J Chem Theory Comput 7:3412–3419
5. Prinz JH et al (2011) Markov models of molecular kinetics: generation and validation. J Chem Phys 134:174105
6. Senne M, Trendelkamp-Schroer B, Mey ASJ, Schütte C, Noé F (2012) EMMA—a software package for Markov model building and analysis. J Chem Theory Comput 8:2223
7. Silva DA, Bowman GR, Sosa-Peinado A, Huang X (2011) A role for both conformational selection and induced fit in ligand binding by the LAO protein. PLoS Comput Biol 7:e1002054
8. Keller B, Daura X, van Gunsteren WF (2010) Comparing geometric and kinetic cluster algorithms for molecular simulation data. J Chem Phys 132:074110
9. Gonzalez T (1985) Clustering to minimize the maximum intercluster distance. Theor Comput Sci 38:293
10. Dasgupta S, Long PM (2005) Performance guarantees for hierarchical clustering. J Comput Syst Sci 70:555
11. Bowman GR, Beauchamp KA, Boxer G, Pande VS (2009) Progress and challenges in the automated construction of Markov state models for full protein systems. J Chem Phys 131:124101
12. Noe F et al (2009) Constructing the equilibrium ensemble of folding pathways from short off-equilibrium simulations. Proc Natl Acad Sci USA 106:19011
13. Tarjan R (1972) Depth-first search and linear graph algorithms. SIAM J Comput 1:146
14. Swope WC, Pitera JW, Suits F (2004) Describing protein folding kinetics by molecular dynamics simulations, 1: theory. J Phys Chem B 108:6571
15. Bowman GR, Geissler PL (2012) Equilibrium fluctuations of a single folded protein reveal a multitude of potential cryptic allosteric sites. Proc Natl Acad Sci USA 109:11681
16. Buchete NV, Hummer G (2008) Coarse master equations for peptide folding dynamics. J Phys Chem B 112:6057

17. Schütte C, Fischer A, Huisinga W, Deuflhard P (1999) A direct approach to conformational dynamics based on hybrid Monte Carlo. J Comput Phys 151:146
18. Deuflhard P, Huisinga W, Fischer A, Schütte C (2000) Identification of almost invariant aggregates in reversible nearly uncoupled Markov chains. Linear Algebra Appl 315:39
19. Deuflhard P, Weber M (2005) Robust Perron cluster analysis in conformation dynamics. Linear Algebra Appl 398:161
20. Noé F, Horenko I, Schütte C, Smith JC (2007) Hierarchical analysis of conformational dynamics in biomolecules: transition networks of metastable states. J Chem Phys 126:155102
21. Bowman GR (2012) Improved coarse-graining of Markov state models via explicit consideration of statistical uncertainty. J Chem Phys 137:134111
22. Yao Y et al (2009) Topological methods for exploring low-density states in biomolecular folding pathways. J Chem Phys 130:144115
23. Singh G, Memoli F, Carlsson G. Mapper: a topological mapping tool for point cloud data. In: Eurographics symposium on point-based graphics
24. Lin J (1991) Divergence measures based on the Shannon entropy. IEEE Trans Inf Theory 37:145
25. Huang X, Bowman GR, Bacallado S, Pande VS (2009) Rapid equilibrium sampling initiated from nonequilibrium data. Proc Natl Acad Sci USA 106:19765
26. Bolhuis PG, Chandler D, Dellago C, Geissler PL (2002) Transition path sampling: throwing ropes over rough mountain passes, in the dark. Annu Rev Phys Chem 53:291
27. Schütte C et al (2011) Markov state models based on milestoning. J Chem Phys 134:204105
28. Levin AM et al (2012) Exploiting a natural conformational switch to engineer an interleukin-2 superkine. Nature 484:529