BIOINFORMATICS

Vol. 00 no. 00 2012
Pages 1–4

Discovering metastable states in MD simulations

Virginia M. Burger,[1,3,*] Arvind Ramanathan,[2,*] Andrej J. Savol,[1,3]
Pratul K. Agarwal[2] and Chakra S. Chennubhotla[3†]

[1]Joint Carnegie Mellon University-University of Pittsburgh Ph.D. Program in Computational Biology, [2]Computational Biology Institute and Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN, 37830, [3]Department of Computational and Systems Biology, University of Pittsburgh, PA 15260.

ABSTRACT

Motivation Protein conformations fluctuate between many metastable states. This is particularly important in the case of folding proteins and in intrinsically disordered proteins. To map the conformational space of proteins, much work has been done developing Markov State Models, which describe the protein's landscape as a set of structurally similar micro states, grouped into sets of kinetically similar macro states. Using these models, the rates of transition between micro and macro states can be explored, and key states and transitions can be identified. Here, we propose a new method for determining macro states based on the bottleneck transitions between metastable states at any time-scale. This method provides new insight into something about how the method is nice in that it is rooted in the time-scales of transitions to intrinsically discover states, and doesn't add in additional theory or concepts (not worried about optimizing a function, which may not be the optimal function, not deriving new theories on the network,..). Also add that it does not require user to determine a number of states, as the number of states is intrinsic to the system at a given time-scale.

We use alanine dipeptide as a toy example to illustrate the method, and then find folding states and rates for a villain headpiece protein.

Contact: chakracs@pitt.edu

1 INTRODUCTION

2 RESULTS: ALANINE DIPEPTIDE

We want to create macro-states using time-scale information. The basic idea behind markov state models is to generate a set of spatially (and thus highly kinetically) similar micro states, and group these states into kinetically similar macro states. Here, we develop a method for grouping the micro states into kinetically similar macro states. Later we show that the macro states are basically independent of the micro states used; as long as the micro states are structurally similar, the discovered macro states will be basically the same at a given time-scale. For now, we will focus on grouping of arbitrary micro states.

2.1 Time-scale based macro states

Once the micro states are defined, we can describe the trajectory as a time-series of micro states, $s = (s_1, \ldots, s_T)$, where T is the number of frames in the trajectory, and s_i is the state at time i, for $i \in \{1, \ldots, T\}$. Let $s_i \in \{1, \ldots, S\}$, where S is the number of micro states. A lag time τ must be determined for which the system is Markovian (discussed below). Given τ, we define the transition count matrix $C_\tau := (c_{ij}^\tau)_{i=1,\ldots,S, j=1,\ldots,S}$, where

$$c_{ij}^\tau := \#\{s(t) = i \wedge s(t+\tau) = j,\, t = 1, \ldots, T\}.$$

We guarantee symmetry by setting $T_\tau := \frac{1}{2}(C_\tau + C_\tau^T)$. A discussion on symmetry, detailed balance, irreducibility, and ergodicity is given below.

Using the transition count matrix, we can view the system of micro states as a network, as in Figure 2. Each node in the graph is defined by a micro state. By normalizing the transition count matrix, we obtain a transition probability matrix M. For this, we let D be the diagonal degree matrix of the network. Thus, $D(i,i)$ is equal to the number of transitions made from node i (the i[th] row sum of T_τ), and $D(i,j) = 0$ for $i \neq j$. Then

$$M_\tau = T_\tau D^{-1}.$$

We discuss the Markov properties of M_τ below. If we let the Markov transition probability between each pair of nodes define the weight of the edge between those nodes in the network, we can imagine performing a random walk on the network. In the network shown in Fig. 2, several clusters are apparent in that many edges connect certain nodes, whereas few edges connect the left side and the right side of the graph. By starting a random walker at any node and letting it move for a few steps, with the probability of traversing an edge defined by the weight of the edge (related to the Markov probability), it is likely that the random walker will stay in the group of nodes strongly connected to that node. Thus, clusters in the network become apparent by performing random walks from each node.

The eigenvectors of M highlight the major clusters in the network (Figure 3). Since M is not symmetric, it is more stable to compute eigenvectors of a similar, symmetric matrix. The normalized graph Laplacian is defined as

$$L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}.$$

Because $M = T_\tau D^{-1} \Rightarrow MD = T_\tau$, it holds that

$$L = D^{-\frac{1}{2}} T D^{-\frac{1}{2}} = D^{-\frac{1}{2}} M D D^{-\frac{1}{2}}$$
$$= D^{\frac{1}{2}} M D^{\frac{1}{2}}.$$

Since L is similar to M, the eigenvalues of L and M are equal, and the eigenvectors of L can be computed as $D^{\frac{1}{2}}x$, if x are the eigenvectors of M. Thus, we will solve for eigenvalues and eigenvectors of the symmetric L instead of M. Transitions in the network occur at a variety of time-scales. Long time-scale

*both authors contributed equally to this work

†to whom correspondence should be addressed

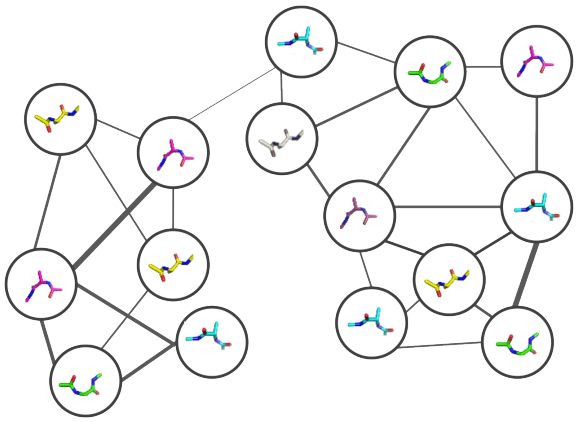

Fig. 1.

transitions occur between clusters that are very separated, for example between the right and left sides of the network, which are only connected by a low probability edge. Short time-scale transitions occur between tightly connected nodes, for example nodes 4,5, and 6, which have are connected by high probability edges. We can group sets of nodes that are highly connected and transition on short time-scales into macro states. Long time-scale transitions occur between macro states. The eigenvalues of the normalized Laplacian are related to the implied time-scales of the Markov network as $\beta_k = \frac{-\tau}{\ln(\lambda_k)}$ (see Appendix), where λ_k is the k^{th} eigenvalue of L.

To cluster the system into temporally related macro states, we analyze the importance of each edge on the time-scales of the system. If changing an edge weight significantly changes the timescales of the system, then that edge is a bottleneck. For example, in Figure 2, changing the weight on the single edge connecting the left and right sides of the network would have a large effect on the network's time-scales. If the edge weight is increased, a random walker could more easily travel between the two halves of the network, shortening the longest time scale of the system. If the edge were made even weaker (or completely removed), then a random walker would almost never be able to travel between the two network halves, lengthening the longest time scale of the system. Thus, this edge is a bottleneck, and by removing it, we separate regions of the network that are separated by a bottleneck.

An overview of the perturbation algorithm is given here. Details are given in Methods.

Algorithm: Perturbation

Input: T_τ, τ,n_e

$n_{\text{cuts}} = 1$.
$n_{\text{iter}} = 0$.
while $n_{\text{cuts}} > 0$ **do**
  $n_{\text{iter}} = n_{\text{iter}} + 1$.
  for each connected component in T_τ **do**
    $n_{\text{nodes}} =$ size of component
    Compute first n_e eigenvectors u_i and eigenvalues λ_i.
    Compute implied time-scales $\beta_i = \frac{-\tau}{\log(|\lambda_i|)}$.
    if $n_{\text{iter}} = 1$ **then**
      $\beta_0 = \beta_{n_e - 1}$.
    end if
    for $i = 0 \rightarrow n_e$ **do**
      if $\beta_i > \beta_0$ **then**
        Compute sensitivity $s_{j,k}$ for all edges $\alpha_{i,j}$, for $j = 1, \ldots, n_{\text{nodes}}$, $k = 1, \ldots, n_{\text{nodes}}$ according to this eigenvector:
        $s_{j,k} = \frac{d \log(\beta_i + \beta_0)}{\alpha_{i,j}}$
        Remove all edges with $s_{j,k} < 0$ in T_τ.
      end if
    end for
  end for
end while
Each resulting connected component in T_τ defines a macrostate.

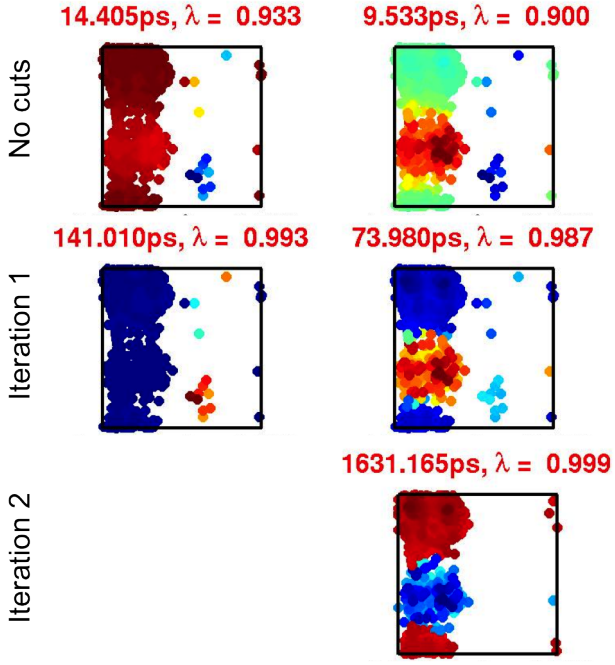

Fig. 2.

Figure 3 demonstrates the changes in timescales after running two iterations of this algorithm.

2.2 Time-scales used for perturbation

The number of eigenvectors used for perturbation is controlled by the desired time-scale of the clustering. In Figure 3, row 1, we see that transitions between the $(\alpha \backslash \beta)$ regions of alanine dipeptide's conformational space and the irregular regions of the space occur at a timescale of 15ps. Transitions between the α and the β regions occur at a time scale of 10ps. If we choose to cut all sensitive edges according to all eigenvectors describing transitions faster than 10ps, we significantly increase the time needed to make these transitions: the time to transition between the $(\alpha \backslash \beta)$ regions and the irregular regions increases to 140ps, and the time to transition between the α and β regions increases to 74ps. We again cut all edges

that are sensitive according to all time-scales slower than 10ps. In iteration two, the $(\alpha \backslash \beta)$ regions and the irregular regions have been completed separated; no more edges exist between these regions, and the time-scale for the transition between the α and β regions has increased to 1630ps, and one more iteration will completely remove edges between these regions.

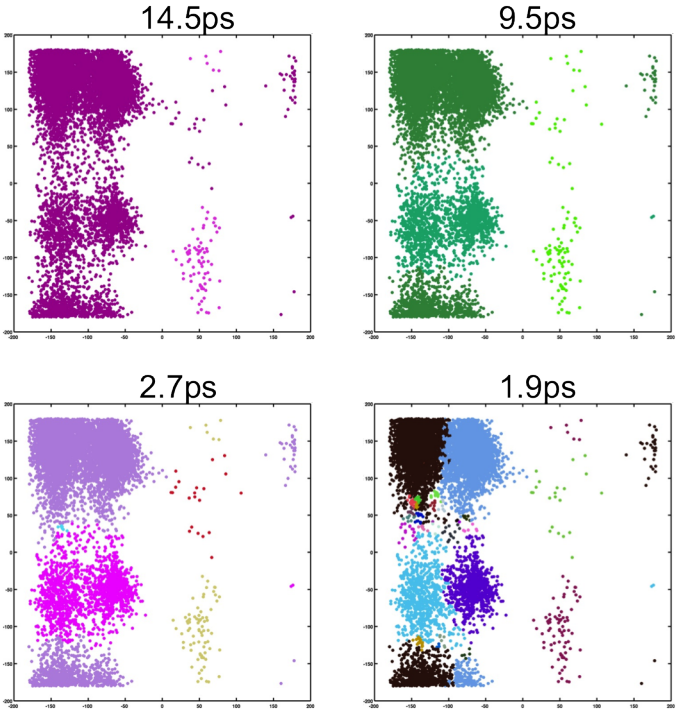

Fig. 3. Using different time-scales for perturbation.

How does this relate to the natural time-scales of alanine dipeptide?

2.3 Microstates

The macro states should be basically independent of the micro states used; any micro state set that guarantees some degree of structural/kinetic homogeneity within a micro state should suffice. To emphasize this we sample a variety of micro state sets, made by (1) gridding the Ramachandran plot, (2) k-means on the dihedral angles, (3) hierarchical clustering of the spatial dihedral similarity network. Figure 1 shows approximately 300 micro states painted on the ramachandran plot using each of these methods. As our temporal clustering merges states with frequent transitions, variations in the micro states are do not cause differences in the resulting temporal network. However, it is important that the micro states are structurally and kinetically very similar, as otherwise unnatural transitions will be included in the network. That is, if two temporally distinct states are included in a single micro state, this micro state will be seen to transition with two very different sets of states. Thus, using many small micro states is better than using few large micro states, as the micro states are more likely to be structurally and kinetically similar as they decrease in size. [V says: *Table scoring the accuracy of the macro states found with each set of micro states against the ground truth states. Important: use several hierarchy levels*]

Symmetry of Count Matrix [V says: *Discussion of how much approximation is being made by symmetricizing the count matrix.*]

detailed balance, irreducibility, and ergodicity [V says: *Show that our micro states fulfill this, at least when the connected component is used. Refer back to that paper. . ..*]

2.4 Influence of Lag Time

[V says: *For one level of the hierarchy (or I guess the best scoring set of micro states), make a table scoring the accuracy of the macro states found with varied lag time. The lag time is important, but is it bad to use lag times longer than the actual lag time? Discussion of computational determination of lag time. Figure: hierarchy level 4, Use different lags!!! Always 6 eigvecs.*]

3 RESULTS: VILLAIN HEADPIECE

4 METHODS

4.1 Edge Perturbation

4.2 Hierarchical Clustering

5 DISCUSSION

6 ACKNOWLEDGEMENTS

AJS was a predoctoral trainee supported by NIH T32 training grant T32 EB009403 as part of the HHMI-NIBIB Interfaces Initiative. PKA acknowledges the support by ORNLs Laboratory Directed Research and Development (LDRD) funds and the computing time allocation from the National Center for Computational Sciences (BIP003). ORNL is managed by UT-Battelle, LLC for the U.S. Department of Energy under Contract No. DEAC05-00OR22725. CSC was partially supported by R01 GM086238.

7 APPENDIX A: DERIVATION OF IMPLIED TIME-SCALES

[V says: *cite JCTC 2011, van gunsteren*].

Given a $(N \times N)$ Markov probability matrix M between N states, which is irreducible, primitive, and satisfies detailed balance with the stationary distribution π, the eigenvectors $\{u_1, \ldots, u_N\}$ form a complete orthonormal basis of $\mathbb{R}^N$. Thus, we can express any vector $v \in \mathbb{R}^N$ as a linear combination of the eigenvectors:

$$v = \sum_{i=1}^{N} \tilde{c}_i u_i.$$

We can relate this form to the eigenvalues $\{\lambda_1, \ldots, \lambda_N\}$ as

$$v = \sum_{i=1}^{N} c_i \lambda_i u_i.$$

Since $M = M(\tau)$ is a Markov probability matrix defined with lag time τ, it holds that

$$p(t + \tau) = M(\tau) p(t)$$

for any probability vector p and that

$$M(n\tau) = M(\tau)^n$$

following the Chapman-Kolmogorov Equation. We know for eigenvalues that $Ax = \lambda x \Rightarrow A^n x = \lambda^n x$, thus if λ_i are eigenvalues of A, then λ_i^n are eigenvalues of A^n.Using these truths, we can write for the probability vector p:

$$\begin{aligned} p(t+n\tau) =& M(n\tau)p(t) \\ &= M^n(\tau)p(t) \\ &= \sum_{i=1}^{N} c_i \lambda_i (t+n\tau) u_i \\ &= \sum_{i=1}^{N} c_i \lambda_i (t)^n u_i. \end{aligned}$$

Thus, we can interpret the eigenvectors as modes of a decay of the probability in p with time $n\tau$. The eigenvalues govern the temporal behavior of the decay.

If we let $t = n\tau$, then

$$\begin{aligned} \lambda_i(t) = \lambda_i(n\tau) &= \lambda_i^n(\tau) \\ &= \lambda_i^{\frac{t}{\tau}}(\tau) = e^{\ln(\lambda_i^{\frac{t}{\tau}}(\tau)))} \\ &= e^{\frac{t}{\tau}\ln(\lambda_i^(\tau)))} = e^{\frac{-\tau}{\beta_i}}, \end{aligned}$$

where we set $\beta_i := \frac{-\tau}{\ln(\lambda_i^(\tau)}$. We refer to β_i as the implied time-scales of the system. They have the same units as τ, which is the unit of the time-step in the molecular dynamics trajectories.

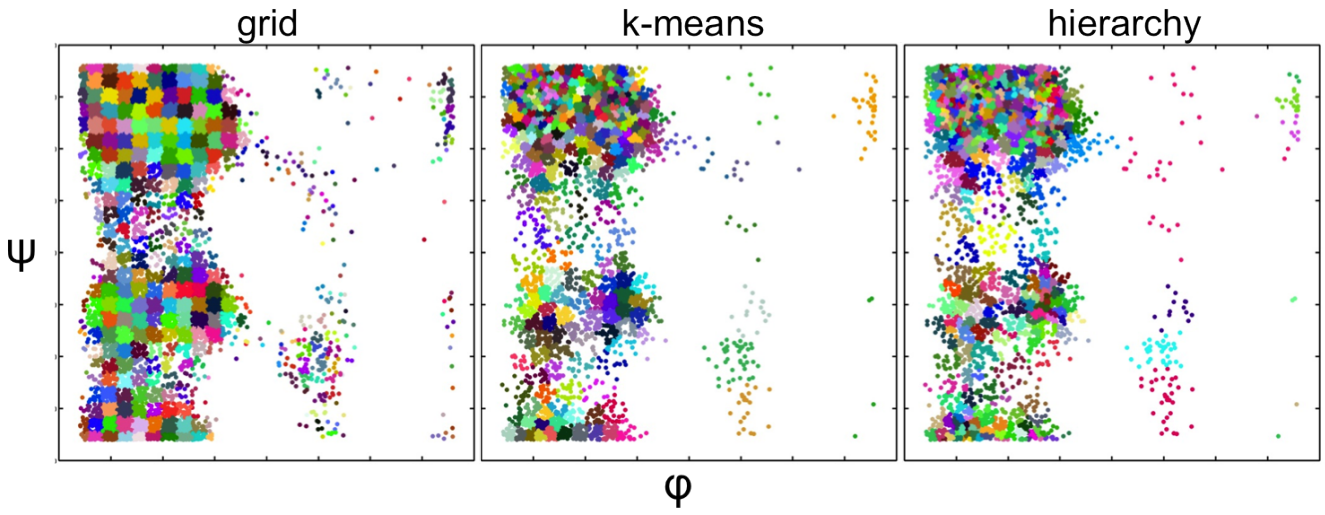

Fig. 4. [V says: *Figure should have a second row showing the resulting macro states, and the time-scales.*]