Methods 49 (2009) 197–201

Contents lists available at ScienceDirect

Methods

journal homepage: www.elsevier.com/locate/ymeth

Using generalized ensemble simulations and Markov state models to identify conformational states

Gregory R. Bowman [a], Xuhui Huang [b], Vijay S. Pande [a,c,*]

[a] Biophysics Program, Stanford University, Stanford, CA 94305, USA
[b] Department of Bioengineering, Stanford University, Stanford, CA 94305, USA
[c] Department of Chemistry, Stanford University, 318 Campus Dr. Rm S295, Stanford, CA 94305, USA

ARTICLE INFO

Article history:
Accepted 17 April 2009
Available online 3 May 2009

Keywords:
Clustering
Molecular dynamics
Thermodynamics
Kinetics

ABSTRACT

Part of understanding a molecule's conformational dynamics is mapping out the dominant metastable, or long lived, states that it occupies. Once identified, the rates for transitioning between these states may then be determined in order to create a complete model of the system's conformational dynamics. Here we describe the use of the MSMBuilder package (now available at http://simtk.org/home/msmbuilder/) to build Markov State Models (MSMs) to identify the metastable states from Generalized Ensemble (GE) simulations, as well as other simulation datasets. Besides building MSMs, the code also includes tools for model evaluation and visualization.

1. Introduction

Molecular Dynamics (MD) and Monte Carlo (MC) computer simulations have the potential to complement experiments by elucidating the chemical details underlying the conformational dynamics of biological macromolecules like proteins and RNA. Such simulations sample a system's free energy landscape, which is characterized by long lived, or metastable, states separated by large free energy barriers. Thus, understanding a system's conformational dynamics can be broken down into two steps: (1) identifying the long lived, or metastable, states visited by the system and (2) determining the rates of transitioning between these states. Unfortunately, it is extremely difficult to adequately sample the conformational space accessible to biomolecules. Furthermore, even if adequate sampling can be achieved, the resulting datasets are often quite large and, therefore, difficult to analyze and interpret.

A popular approach to the first step is to use Generalized Ensemble (GE) algorithms [1–5] to sample the accessible space and then to generate projections of the free energy landscape onto some set of order parameters to identify the dominant thermodynamic states [6–9]. GE algorithms, such as the Replica Exchange Method (REM) [2,3] and Simulated Tempering (ST) [4,5], achieve broad sampling at the temperature of interest by performing a random walk in temperature space. Broad sampling is possible because an energy barrier that is difficult to cross at the temperature of interest may be crossed more easily at higher temperatures. GE algorithms also maintain canonical sampling at every temperature. Thus, they are a suitable way to sample the accessible space.

Projections of the free energy landscape onto a few order parameters are frequently used to make sense of the resulting dataset [6–9]. Such projections may be meaningful if an appropriate set of order parameters is chosen; however, this is quite difficult so there is always the danger of being misled by projections because meaningful information along other order parameters may be completely lost [10,11]. For example, structures that fall within the same basin in some projection may have little structural or kinetic similarity. Thus, choosing a representative conformation for that basin may be impossible.

Clustering methods, on the other hand, do not have these issues because the dominant order parameters do not need to be specified in advance. However, most clustering algorithms group conformations together based solely on their structural similarity [12,13], so they may fail to capture important kinetic properties. To illustrate the importance of integrating kinetic information into the clustering of simulation trajectories, one can imagine two people standing on either side of a wall. Geometrically these two individuals may be very close but kinetically speaking it could be extremely difficult for one to get to the other. Similarly, two conformations from a simulation dataset may be geometrically close but kinetically distant and, therefore, a clustering based solely on a geometric criterion would be inadequate for describing the system's dynamics.

Here we describe the use of Markov State Models (MSMs) to identify metastable states in GE datasets, though we note that the MSMBuilder package we introduce to build MSMs may be

* Corresponding author. Address: Department of Chemistry, Stanford University, 318 Campus Dr. Rm S295, Stanford, CA 94305, USA. Fax: +1 650 724 4021.
E-mail address: pande@stanford.edu (V.S. Pande).

1046-2023/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.ymeth.2009.04.013

applied to any simulation dataset. An MSM may be thought of as a form of clustering that incorporates kinetic information by grouping conformations that can interconvert rapidly into the same state and conformations that cannot interconvert rapidly into different states [11,14–18]. Thus, conformations in the same metastable state, which may be thought of as a large free energy basin, will be grouped together while conformations separated by large free energy barriers will not.

A biomolecular folding free energy landscape may be thought of as a hierarchy of basins [19,20]. Since larger basins may contain numerous smaller local minima our use of the phrase free energy basin above is somewhat ambiguous. To determine what constitutes a distinct free energy basin an MSM may be represented as a transition probability matrix where the entry at row i and column j gives the probability of transitioning from state i to state j during a time Δt, called the lag time. Based on this matrix one may obtain a series of implied timescales for transitioning between various regions of phase space and use this information to determine an appropriate number of metastable states, as explained below. The number of metastable states to be constructed controls the resolution of the model by determining how large a barrier must be in order to divide phase space into multiple states.

In the past, MSMs have generally been used to model kinetics and, therefore, have been built from constant temperature data. For example, MSMs have been used to model numerous small systems [16,21,22] and a few larger ones [23,24]. Since GE simulations perform a random walk in temperature space they do not have physical kinetics. However, GE simulations contain the desired canonical ensemble and therefore the desired free energy barriers. These barriers may be flattened or distorted at higher temperatures but the barriers at the temperature of interest should still be sufficient to provide the desired separation of timescales. That is, fast intrastate transitions and slower interstate transitions. Thus, the pseudo-kinetics of GE simulations are still sufficient to identify the dominant metastable states.

In the following sections we describe the use of the MSMBuilder package (now available at http://simtk.org/home/msmbuilder/) to identify the dominant metastable states in GE datasets, though we note the method may be applied as is to datasets generated with other algorithms and is easily extensible to completely different problems. There are four major steps in the procedure: (1) dividing the data into small sets called microstates based on their structural similarity, (2) lumping kinetically related microstates together into metastable states (also called macrostates), (3) extracting representative conformations for each state, and optionally (4) calculating populations of each state to judge convergence. Steps 1–3 are depicted schematically in Fig. 1. The conformations extracted with this method represent the space explored by the system and thus give insights into its dynamics. The pseudo-kinetics of the GE simulations may give some indication of the connectivity of these states but cannot give conclusive results due to the random walk in temperature space. However, this method may serve as a basis for obtaining both accurate thermodynamics and kinetics (Huang et al., in preparation).

2. Description of method

2.1. Dividing the data into microstates

The first step in building an MSM is to divide the data into thousands of microstates based on their structural similarity [15]. For conformational dynamics we measure structural similarity by the RMSD for some subset of the atoms. While the RMSD may not be very meaningful for large distances, it does have a kinetic interpretation for small distances. That is, conformations with very small RMSDs should be able to interconvert rapidly. Thus, if a microstate is small enough that every member has a very small RMSD to every other member then one may assume that their structural similarity implies a kinetic similarity.

However, one must also take care not to generate microstates that are too small because it is important to see a sufficient number of transitions between them. For example, if every conformation were put into its own microstate no pair of trajectories would ever visit the same microstate. Thus, the most meaningful grouping of microstates would be to group every conformation in the same trajectory together and no new insight would be gained.

One method for determining an appropriate size for each microstate is to measure the average RMSD between every pair of temporally adjacent conformations in each trajectory and to ensure that the diameter of each microstate is no more than this value (Sun et al., in preparation). preparation). Thus, any pair of conformations within a given microstate will tend to be within one MD step of each other. However, this method may be overly stringent. We have found that using microstates with an all-heavy-atom RMSD radius of about 3.0 Å allows us to capture the true equilibrium distribution for an eight nucleotide RNA hairpin. Preliminary work in our lab shows that radii on the order of 2–2.5 Å seem more appropriate for protein systems.

One can use the `doFastGromacsClustering` executable provided by the Clusterer component of the MSMBuilder package to divide a dataset into microstates. At present the Clusterer code is capable of using an approximation of the k-centers clustering algorithm [25,26] to divide simulation datasets generated with the Gromacs software package [27] into some number of microstates. However, it is written in object oriented C++ code so it is straight forward to add new clustering algorithms, data types to cluster, distance metrics, and other components.

The approximate k-centers clustering algorithm was chosen as the default clustering method because it is deterministic, simple, fast, and creates clusters with approximately equal radii [26]. The algorithm works as follows: (1) every point is initially infinitely far from any cluster center, (2) choose an arbitrary point

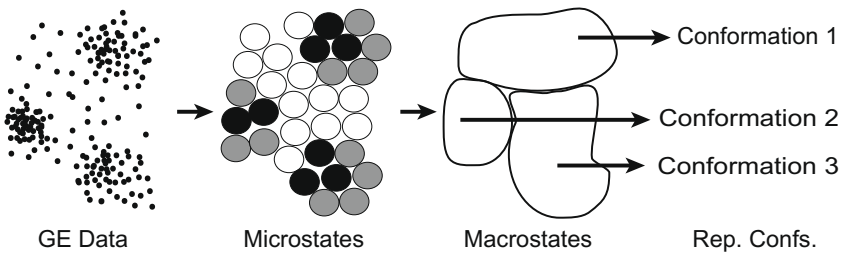

Fig. 1. Schematic of the steps required for building an MSM and obtaining representative conformations for each state. First, GE data represented by points are grouped into microstates represented by circles, with darker circles for more highly populated microstates. Kinetically related microstates are then lumped together into macrostates, or metastable states, represented by amorphous shapes. Finally, representative conformations are obtained by extracting the most probable conformation from each macrostate.

as the first cluster center, (3) compute the distance between every point and the new cluster center, (4) assign points to this new cluster center if they are closer to it than the cluster center they are currently assigned to, (5) declare the point that is furthest from every cluster center to be the next new cluster center, and (6) repeat steps 2–5 until the desired number of clusters have been generated. Thus, the algorithm has complexity O(kN) where k is the number of clusters to be generated and N is the number of data points to be clustered. An order of magnitude speedup is also made possible by using the triangle inequality to avoid unnecessary distance computations (Sun et al., in preparation). This fast version of the algorithm is used by default, though the original version described above is also available. Besides the cluster definitions, this program also gives the radius of each microstate and the average and standard deviation of the RMSD from every member of the microstate to the cluster center.

The arbitrary starting point used by this approximate k-center clustering algorithm would be of some concern for small k or if the microstates were our primary interest. However, we have found that the clustering results are insensitive to the starting point for large k (e.g. $k > 1000$). In addition, we are mainly concerned with the macrostates generated by lumping kinetically related microstates together. The lumping algorithm described in the next section is fairly insensitive to the exact boundaries between microstates as long as each microstate is sufficiently small, so the arbitrary starting point is acceptable for building MSMs.

An attractive feature of this approximate k-centers algorithm is that it yields clusters of approximately equal volume (as judged by using the maximal distance between the cluster center and any other point in the cluster as the radius of a sphere) [26]. This property is of value because it means that the population of a cluster is approximately proportional to its density in phase space. However, we note that exploiting this interpretation requires some caution as it is unclear how to compute exact volumes in a high dimensional phase space and, therefore, difficult to measure densities in phase space precisely. Regardless, this property also allows the boundaries between metastable states to be well-resolved. Clustering algorithms that do not have this guarantee may create large clusters in sparse regions of phase space and small clusters in dense regions. The large clusters in sparse regions of phase space are prone to violate the assumption that conformations within a microstate are kinetically related. Therefore, various conformations in the microstate may be most kinetically related to different metastable states, in which case it will be unclear which macrostate to group the microstate with.

2.2. Lumping microstates into macrostates

Conceivably, one could extract a representative conformation for each microstate to get an idea of the conformational space explored by the system of interest. However, this would only be a slight improvement upon examining the raw data itself. Instead, it is valuable to lump kinetically related microstates together into metastable states, also called macrostates. The tools for lumping together microstates, as well as for extracting representative conformations and determining state populations, may be found in the PythonTools component of the MSMBuilder package.

The first step in generating a set of macrostates is to determine how many of them to create [15]. This task may be accomplished with the `BuildMSMsAsVaryLagTime.py` script. This script builds a microstate MSM for each of a series of lag times. A microstate MSM is just a transition probability matrix where the entry in row i and column j is the probability that a simulation will be in microstate j at time $t + \Delta t$ given that it was in state i at time t. A series of implied timescales are then calculated and printed to a file for each microstate MSM based on the eigenvalues of the transition probability matrix. These implied timescales correspond to the timescales for transitioning between different sets of microstates. An appropriate number of macrostates to build can be determined based on the location of the major gap in the implied timescales, which should correspond to the largest separation of timescales within the system. The implied timescales for multiple lag times are examined because the location of this gap is normally sensitive to the lag time. Ideally the implied timescales will level out as the lag time increases [17] and obvious gaps that are robust with respect to the lag time will be apparent, as indicated in Fig. 2. An appropriate number of macrostates is then one more than the number of implied timescales above the major gap [11,15]. In non-ideal cases the number of implied timescales above the gap will not level off. In such cases we recommend erring on the side of having too many macrostates rather than too few. If too many macrostates are generated then some of the representative conformations may be redundant (only separated by small barriers), whereas if too few are constructed important regions of phase space may not be identified.

A macrostate MSM with the appropriate number of states may then be built using the `BuildMacroMSM.py` script. First, this script uses the Perron Cluster Cluster Analysis (PCCA) algorithm [28,29] to lump together kinetically related microstates. The PCCA algorithm identifies kinetic relationships based on the eigenvalue/eigenvector structure of the microstate MSM and will not be described in detail here. This initial lumping is then refined using simulated annealing to maximize the metastability [15], which is defined as

$$Q = \sum_{i=1}^{N} T(i,i) \tag{1}$$

where N is the number of macrostates and T is the macrostate MSM transition probability matrix. In words, the metastability is the sum of the self-transition probabilities of each macrostate. Thus, the metastability may range from 0 to N. Maximizing the metastability is a heuristic for maximizing the separation of timescales [15]. During each simulated annealing step a randomly selected microstate is reassigned to a randomly selected macrostate, the resulting change

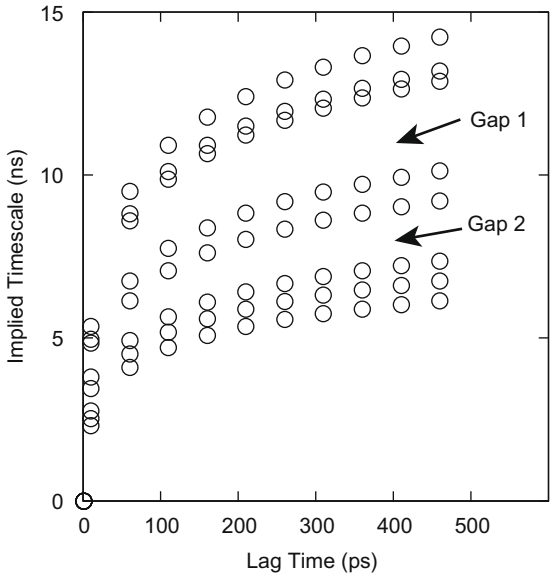

Fig. 2. Implied timescales as a function of the lag time. There are two probable gaps in the implied timescales. If gap one were selected then a macrostate MSM with four states would be constructed whereas if gap two were selected a higher resolution MSM with six states would be constructed.

in metastability is calculated, and the move is either accepted or rejected based on the Metropolis criterion.

We recommend using a lag time of one step to build the MSM to maximize the use of all the data. The resulting state definitions and a longer lag time may then be used to obtain populations and transition rates. A lag time within the implied timescale gap should yield a strongly Markovian model. That is, one with a sufficiently large separation of timescales that the assumption that the state at time $t + \Delta t$ depends only on the state at time t is valid.

The main outputs of the `BuildMacroMSM.py` script are a mapping from microstates to macrostates and the metastability of this lumping. The mapping from microstates to macrostates may be used to determine which macrostate each data point is in using the `WriteMacroAssignments.py` script or the `doFastGromacsAssign` program. In general, the `WriteMacroAssignments.py` script should be used as it is faster. Both methods allow the user to specify a temperature range and will only print out assignments for conformations within this range. This feature is useful for calculating populations of states at a given temperature. The mapping may also be used by the `getMacroStateCenters` program to get information about each macrostate, such as the most geometrically central microstate and the average and the standard deviation of the RMSD between that microstate's center and the center of every other microstate in that macrostate. Such information is useful for getting an idea of the size of each macrostate.

2.3. Extracting representative conformations

There are a number of ways of extracting representative conformations for each macrostate. A simple way of getting a single conformation is to use the `getMacroStateCenters` program as discussed above. However, one must remember that conformations selected in this manner represent the geometric center of each macrostate and not necessarily the most probable member of each macrostate.

To understand the distribution of conformations in each macrostate one may identify the central conformation of each microstate in a given macrostate using the `GetMicroCentersByMacroState.py` script. The conformations for a given macrostate may then be overlaid in a viewer for visual analysis. Such an approach may be cumbersome if there are too many microstates in each macrostate. One alternative is to randomly select a reduced number of conformations from each macrostate using the `GetRandomConfsFromEachState.py` script. A major shortcoming of these methods is that they select conformations with a more or less uniform distribution across the macrostate.

Probably the best way of extracting representative conformations is to use the `GetDensityInfo.py` script. This script outputs a list of the microstates in each macrostate ordered from densest to sparsest. That is, the most probable to the least probable. Any number of the most probable structures in a given macrostate may then be selected and overlaid in a viewer to get an idea of the distribution of conformations within the state.

2.4. Judging convergence

Unfortunately there is no analytic way of checking that a single set of simulations has explored the entire accessible space for a given system and, therefore, yielded representative conformations that accurately describe the conformational dynamics. To the best of our knowledge, the most effective way to ensure that the entire space has been explored is to run two distinct sets of simulations started from very different initial configurations. The populations for each state may then be calculated for each dataset. If they agree then one can be relatively sure that the entire space has been explored because the thermodynamics found are independent of the starting conformation.

One practical consideration is that the same state definition must be used for both datasets because it is unclear how to compare different MSMs. A common state definition may be obtained by building a single MSM based on both datasets. The `WriteMacroAssignments.py` script or `doFastGromacsAssign` program may then be used to independently assign each dataset to this common state definition, preferably restricting the assignments to the temperature range of interest for GE datasets, so that the population of each state may be determined. Of course, due to the stochastic nature of conformational dynamics the two sets of populations are unlikely to agree exactly. To make a valid comparison the `GetMacroMSMPopStats.py` script may be used to obtain error bars on the populations from each dataset. This script uses a bootstrapping algorithm to approximate the variation in the populations. If the populations agree within error then the two simulations may be considered to have converged to the true equilibrium distribution and one may be relatively sure that the entire accessible space has been explored. Thus, the conformations extracted in step 3 will provide an accurate depiction of the conformational dynamics of the system.

3. Concluding remarks

Using the MSMBuilder to analyze GE simulations and other datasets will allow researchers to quickly map out the conformational space explored by biological macromolecules like RNA, which is the first step to understanding conformational dynamics. The MSMBuilder may also be used to determine the rates of transitioning between states in microcanonical and canonical simulations, resulting in a complete Markov state model for the system's conformational dynamics. While other algorithms for building MSMs exist [15], they are not likely to provide much improvement for analyzing GE datasets due to the distortion resulting from high temperature data. The highly extensible object oriented design of the code should allow such algorithms to be incorporated easily for use with other datasets though. Incorporating other data types, clustering methods, distance metrics, and analysis tools should also be straight forward. In particular, this software serves as a foundation for automating adaptive sampling algorithms [30], which promise to allow the maximal use of one's computing resources by focusing sampling on regions of uncertainty. Finally, the results of applying this method to GE datasets may be used as a basis for determining the rates of transitioning between states (Huang et al., in preparation), thereby giving a complete picture of a system's dynamics.

Acknowledgments

Thanks to Drs. John Chodera and Jian Sun for their insight into MSMs and approximate *k*-centers clustering, respectively. G.B. is supported by the NSF Graduate Research Fellowship Program. X.H. is supported by the National Institutes of Health through the NIH Roadmap for Medical Research, Grant No. U54 GM072970 and would also like to thank Professor Michael Levitt for his support. This work was also supported by NIH R01-GM062868 and NIH P01-GM066275.

References

[1] A. Mitsutake, Y. Sugita, Y. Okamoto, Biopolymers 60 (2001) 96–123.
[2] U.H. Hansmann, Y. Okamoto, Curr. Opin. Struct. Biol. 9 (1999) 177–183.
[3] Y. Sugita, Y. Okamoto, Chem. Phys. Lett. 314 (1999) 141–151.
[4] A.P. Lyubartsev, A.A. Martsinovski, S.V. Shevkunov, P.N. Vorontsov-Velyaminov, J. Chem. Phys. 96 (1992) 1776–1783.
[5] E. Marinari, G. Parisi, Eur. Lett. 19 (1992) 451–458.

[6] R. Zhou, B.J. Berne, R. Germain, Proc. Natl. Acad. Sci. USA 98 (2001) 14931–14936.
[7] Y.M. Rhee, V.S. Pande, Biophys. J. 84 (2003) 775–786.
[8] H. Nymeyer, A.E. Garcia, Proc. Natl. Acad. Sci. USA 100 (2003) 13934–13939.
[9] R. Zhou, Proc. Natl. Acad. Sci. USA 100 (2003) 13280–13285.
[10] S.V. Krivov, M. Karplus, Proc. Natl. Acad. Sci. USA 101 (2004) 14766–14770.
[11] F. Noe, S. Fischer, Curr. Opin. Struct. Biol. 18 (2008) 154–162.
[12] M.E. Karpen, D.J. Tobias, C.L. Brooks 3rd, Biochemistry 32 (1993) 412–420.
[13] J.Y. Shao, S.W. Tanner, N. Thompson, T.E. Cheatham, J. Chem. Theory Comput. 3 (2007) 2312–2334.
[14] C. Schutte, Conformational Dynamics: Modeling, Theory, Algorithm, and Application to Biomolecules. Habilitation Thesis, Department of Mathematics and Computer Science, Freie Universitat, Berlin, 1999, pp. 1–139.
[15] J.D. Chodera, N. Singhal, V.S. Pande, K.A. Dill, W.C. Swope, J. Chem. Phys. 126 (2007) 155101.
[16] N.V. Buchete, G. Hummer, J. Phys. Chem. B 112 (2008) 6057–6069.
[17] W.C. Swope, J.W. Pitera, F. Suits, J. Phys. Chem. B 108 (2004) 6571–6581.
[18] S. Sriraman, L.G. Kevrekidis, G. Hummer, J. Phys. Chem. B 109 (2005) 6479–6484.
[19] H. Frauenfelder, S.G. Sligar, P.G. Wolynes, Science 254 (1991) 1598–1603.
[20] W.Y. Yang, M. Gruebele, J. Am. Chem. Soc. 126 (2004) 7758–7759.
[21] N. Singhal, C. Snow, V.S. Pande, J. Chem. Phys. 121 (2004) 415–425.
[22] S. Elmer, S. Park, V.S. Pande, J. Chem. Phys. 122 (2005) 124908.
[23] G. Jayachandran, V. Vishal, V.S. Pande, J. Chem. Phys. 124 (2006) 164902.
[24] N.W. Kelley, V. Vishal, R. Jaykumar, C. Liu, T. Pray, G. Krafft, V.S. Pande, J. Chem. Phys. 129 (2008) 214707.
[25] T. Gonzalez, Theor. Comput. Sci. 38 (1985) 293–306.
[26] S. Dasgupta, P.M. Long, J. Comput. Syst. Sci. 70 (2005) 555–569.
[27] E. Lindahl, B. Hess, D. van der Spoel, J. Mol. Model. 7 (2001) 306–317.
[28] P. Deuflhard, M. Weber, Linear Algebra Appl. 398 (2005) 161–184.
[29] P. Deuflhard, W. Huisinga, A. Fischer, C. Schütte, Linear Algebra Appl. 315 (2000) 39–59.
[30] N.S. Hinrichs, V.S. Pande, J. Chem. Phys. 126 (2007) 244101.