Using Spectral Clustering to Sample Molecular States and Pathways

Surl-Hee Ahn[1, a)] and Johannes Birgmeier[2, b)]
[1)] *Chemistry Department, Stanford University, Stanford, California 94305, USA*
[2)] *Computer Science Department, Stanford University, Stanford, California 94305, USA*

Molecular dynamics (MD) simulations offer a way to explore the conformational state space of large, biologically relevant molecules. Our sampling method, called "concurrent adaptive sampling" (CAS), utilizes MD simulations by letting a number of "walkers" adaptively explore the state space in consecutive steps. Walkers start in one conformational state, execute a short MD simulation and thus end up in a different state. One of the properties of CAS algorithm is that the number of walkers explodes quickly as the full state space of the biomolecule is explored. Hence, we use a technique called "spectral clustering" in order to cluster similar walkers whenever there are too many walkers in the state space. After clustering, multiple similar walkers are replaced by a lower number of walkers within each cluster. This results in a large speedup of the sampling algorithm.

I. INTRODUCTION

Exploring the state space of biomolecules using MD simulations is a relevant problem in areas such as drug design. But how does one exactly start to explore the state space of a biomolecule? We first look at few important degrees of freedom that the biomolecule has, dubbed as collective variables, and use them to describe the state space. For instance, when we study amino acids, we often look at the two dihedral angles ϕ and ψ and plot their Ramachandran plots, which tell us the possible conformations and their probabilities. See Fig. 1 as an example.

However, existing simulation methods are either too slow or too inaccurate to deliver valuable information about large molecules. Accelerating methods that reveal information about the state space while preserving accuracy is therefore an important problem. Hence, our goal is to accelerate one sampling method called the concurrent adaptive sampling (CAS) using clustering techniques, both of which will be described in detail in Section II. Specifically, we use the CAS algorithm to explore the state space of penta-alanine and clustering techniques to accelerate the exploration process. The CAS algorithm adaptively creates microstates corresponding to certain conformations by running many short simulations, or "walkers." Since there will be numerous microstates after the simulation runs for a while, we use clustering techniques to cluster the microstates into macrostates and reduce "redundant" microstates within a macrostate. **In summary, the input to our algorithm is the set of microstates created from the CAS algorithm simulations. We then use spectral clustering to output predicted macrostates, or clusters of microstates, and reduce the overall number of microstates and associated walkers for computational efficiency.**

We describe a simplified version of the CAS algorithm in in Section II A because the project does not make sense without its description.

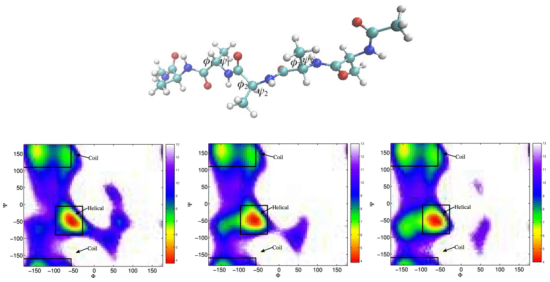

FIG. 1. Structure of penta-alanine and its Ramachandran plots based on its three middle ϕ and ψ dihedral angle pairs. The most probable to least probable conformations are colored from red to white as shown in the colormap. Figure taken from Ref. 1.

II. METHODS

A. Concurrent adaptive sampling

The basic outline of the CAS algorithm is as follows:

1. A circular microstate or "ball" is a collection of similar molecular states of the biomolecule. The "ball" is named so because it contains all states that are within some fixed radius r. Also, the ball's center's values are equal to the collective variables' values. In our case, the collective variables are the six dihedral angles, so our "ball" is a six dimensional sphere with a center equal to the dihedral angle values. As we describe the algorithm, it will become clear that balls are indeed microstates that help us structure the (infinitely large) state space. All microstates have the same radius r.

2. A "walker" is an abstract notion of a tiny worker that starts in a particular conformational state, executes a very short MD simulation of length τ from that state and ends up in a different conformational

[a)] Electronic mail: sahn1@stanford.edu
[b)] Electronic mail: jbirgmei@stanford.edu

state. All walkers reside in balls, which will be explained later.

3. A walker also has a "weight" or probability of ending up in the state that the walker is currently in. Hence, the sum of the weights of all currently existing walkers is always 1.

4. To start the CAS algorithm, place a target number of walkers n_w in the initial state of a given biomolecule. For example, if n_w is set to be 10, each walker in the initial state will have a weight of 1 since the sum of all weights is maintained to be 1. This target number of walkers n_w will be used for subsequent steps, which will be explained later.

5. To start sampling, loop through the following ad infinitum or timeout:

   (a) For each walker that currently exists: execute a short MD simulation of length τ such that the walker "walks" from its current state to the next state.

   (b) After the walkers have finished "walking,": for each walker, check if they ended up within an existing ball. For any walker that resides outside any ball, create a new ball around the walker. That is, create a ball centered around the first walker and then check to see if the next walker lie within the ball. If not, then create a new ball for the next walker and so on.

   (c) After the ball construction, loop through each ball and check to see if there exists a target number of walkers n_w in each ball. If there are more or less walkers than n_w, then merge and/or split walkers to end up with n_w walkers. Each walker will end up with weight equal to the mean weight of the walkers in the ball and this method is called "resampling"[2]. It is essential for the sampling algorithm to maintain a target number of walkers n_w in each ball to constantly observe visited states irrespective of their energy barriers.

B. Spectral clustering

As the number of balls grows, the number of walkers or MD simulations also increases and computing one step of the simulation becomes computationally expensive. Hence, we perform a clustering of balls after the number of balls hits a certain threshold. Initially, we create a transition matrix of the existing balls and calculate their transition probabilities using their weights and previous and current coordinates. Then we perform eigendecomposition on the transition matrix of the balls and obtain the normalized second eigenvector, which is the second eigenvector normalized by the first eigenvector, or equilibrium weights of the balls. The second eigenvector represents the probability changes of the balls in the slowest non-stationary process. We then use the normalized second eigenvector to cluster balls using k-means, and this is done because we would like to cluster balls that are similar dynamically. This is described more in detail in Section IV.

After clustering, we downsample the number of walkers in each cluster in the following way: each cluster receives the same, fixed number of walkers n_{wc}. This number n_{wc} is set to be significantly lower than the average number of walkers in each cluster. If there are more than n_{wc} balls in any cluster, then only n_{wc} of the balls are used, which are picked randomly, so that one walker lies in each ball and the rest are deleted.

III. RELATED WORK

Developing accelerated sampling methods for MD has been an ongoing research problem and several methods currently exist[3–5]. However, most of these methods alter the real kinetics of the system and therefore, we are unable to find statistically correct pathways and intermediates from using these methods.

Hence, different methods have been developed to preserve real kinetics of the system. One method is the weighted ensemble (WE) method developed by Huber and Kim, which is the main method that the CAS algorithm is built on top of[6]. In particular, the CAS algorithm is one of the adaptive versions of the WE method[6–10].

Another method is building Markov state models (MSMs), which divides up the state space into small microstates and runs a large number of short trajectories to compute transition probabilities between microstates, overall slow reaction rates, and other kinetic and thermodynamic properties[11]. However, there is no technique to efficiently build an MSM a priori - the microstates need to be small and the lag time τ needs to be long so that transitions are Markovian but not too long so that short-lived transitions are not captured. In MSMs, geometric clustering methods are used to build an MSM and Perron cluster cluster analysis (PCCA) is used to identify metastable states of the system. Our clustering algorithm, however, aims to cluster microstates based on their dynamic similarity rather than geometric or energetic similarity.

IV. DATASET AND FEATURES

We always cluster **balls**, not walkers or any other entity in the system. The adaptive sampling algorithm previously described generates the dataset that we use for clustering balls. The input to the sampling algorithm is very small: it consists only of the initial state(s) of the molecule we're simulating (i.e., penta-alanine) for the

sampling method. However, by letting the walkers explore the state space, in which every state is represented by six different dihedral angles, the data needed for clustering is generated. This data consists of:

- The coordinates of a large number of balls after running adaptive sampling for a number of steps.
- The transition matrix T, which is calculated from the transitions of walkers between balls in two consecutive steps. E.g., if the net flux between balls 150 and 160 consisted of walkers with weight 0.02, then the entry $T_{ij} = 0.02$. This transition matrix corresponds to the "graph Laplacian matrix" L described in[12,13].

The largest eigenvalue of the transition matrix is always 1. Its corresponding eigenvector reveals the distribution of weights among balls such that after performing any number of simulation time steps, the distribution stays exactly the same. This is called the *equilibrium state*.

The second largest eigenvalue of the transition matrix is less than but close to 1. Its corresponding eigenvector reveals the distribution of weight changes among balls such that after performing a large number of simulation timesteps, the system will converge towards the *first metastable state*. Because we are interested in preserving as much information as possible about the convergence towards the first metastable state, we cluster based on the second eigenvector values. However, since the second eigenvector values differ in magnitudes with each other, we "normalized" the second eigenvector values by dividing them by their corresponding first eigenvector values. Hence, the normalized second eigenvector is one of our features. Clustering states based on values in the eigenvectors is classical spectral clustering.

After performing spectral clustering using the normalized second eigenvector, we discovered that very distant states in the state space would often get clustered together simply because they have similar normalized second eigenvector values. Therefore, we also tried expanding our feature set to cluster based on both ball coordinate information and the normalized second eigenvector. We present results based on clustering only with the normalized second eigenvector, and clustering with both normalized second eigenvector and ball coordinate information.

Our final feature set therefore consists of:

- The six ball coordinates that form the center of a ball, for balls 1 to n, resulting in a matrix of dimensionality $n \times 6$.
- The normalized second eigenvector, of dimensionality $n \times 1$.

The full feature matrix therefore has the dimensionality $n \times 7$. This feature matrix is used in k-means clustering as described in Section II.

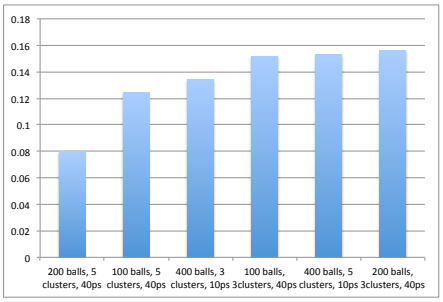

FIG. 2. Average Dunn indices for clustering based on normalized second eigenvector and ball coordinate information in various configurations. Higher is better. For clustering at a 400 balls threshold, the 40 ps timestep led to a timeout after 48 hours before clustering 400 balls was reached, so we repeated the experiment with a 10 ps timestep. Clustering at 200 balls threshold with 5 clusters performed worst, but clustering at 200 balls threshold with 3 clusters performed best. We do not show the silhouette scores separately because they agree with the Dunn index.

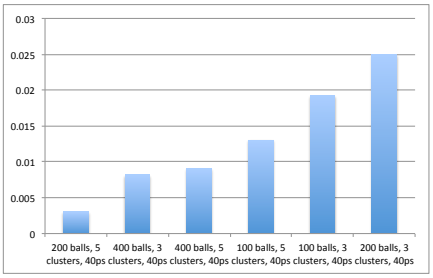

FIG. 3. Average Dunn indices for clustering based on normalized second eigenvector only in various configurations. Higher is better. Surprisingly, the configurations that give the best and worst performance are the same as in Fig. 2; this holds even though clustering was performed on different simulation replicas. Apart from that, it is hard to see trends in clustering performance.

V. EXPERIMENTS/RESULTS/DISCUSSION

A. Performance

Running the sampling algorithm for 48 hours in various configurations, which we did repeatedly during development, typically resulted in a timeout at approximately step 40-50.

B. Clustering and outliers

To assess the performance of the clustering methods, we implemented both the Dunn index and silhouette scores for the clusters. Fig. 2 and Fig. 3 contain a comparison of the clustering performance in various configurations based on the Dunn index.

We also used the silhouette score during spectral clus-

tering to determine the quality of clustering. It turned out that sometimes the clustering result was heavily influenced by a few outliers. In these cases, the outliers would end up in a couple of tiny clusters, and a single huge cluster would contain the bulk of balls. We observed that when this case occurred, the average silhouette score was always higher than 0.8. Using this as a threshold, we sought to mitigate the outlier problem in the following ways:

- We performed outlier detection using an Elliptic Envelope with the goal of removing 5% of the data as outliers. Afterwards, we would perform spectral clustering as usual on the inliers. The SKLearn implementation relies on the robust estimation of a non-singular covariance matrix of the data. However, the covariance matrix used for outlier detection often turned out to be singular, so we could not perform outlier detection. Failing to remove outliers and proceeding with the highly imbalanced clusters degenerated the performance of the CAS algorithm with spectral clustering.
- Therefore, we turned to a simpler solution whenever outlier detection failed: Forego spectral clustering for this step even though the number of balls is above the pre-set threshold. Instead, wait for one more timestep and perform spectral clustering afterwards if the clustering results are not influenced as heavily by outliers anymore. This strategy was mostly successful, even though the additional timestep often took a lot of time to compute.

C. A view on the clusters

In all of the following figures, the colors correspond to the different clusters. We show only clustering results after using k-means with three or five clusters; hence, there are three or five colors in each plot. When showing clustering results, we are always showing a plot of the first two dihedral angles ϕ and ψ (a Ramachandran plot), except for the PCA plot.

Clustering based on the normalized second eigenvector resulted in a "messy" clustering, as seen in Fig. 4. However, it is important to keep in mind that all states in one cluster share similar dynamical properties, even if those are not readily visible. Clustering based on both ball coordinate information and normalized second eigenvector results in a somewhat more sensible clustering as seen in Fig. 5. In particular, it is visible that all states in the upper left corner of the Ramachandran plot are somewhat similar in both dihedral angles and normalized second eigenvector values.

One of the reasons that the clustering results still look very jumbled to the human eye is that we can show only two of the seven dimensions used for clustering in one plot. When clustering is based on the first two ball coordinates and normalized second eigenvector values, the

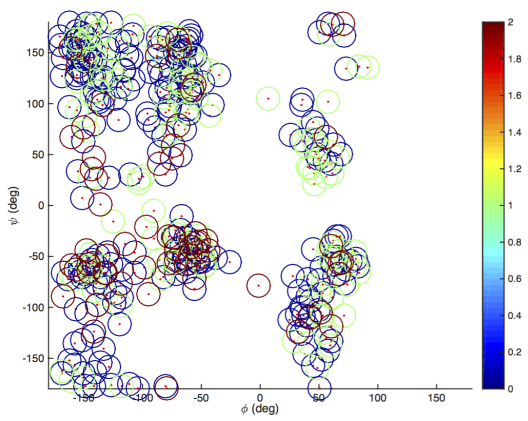

FIG. 4. Clustering results based on the normalized second eigenvector using k-means with five clusters.

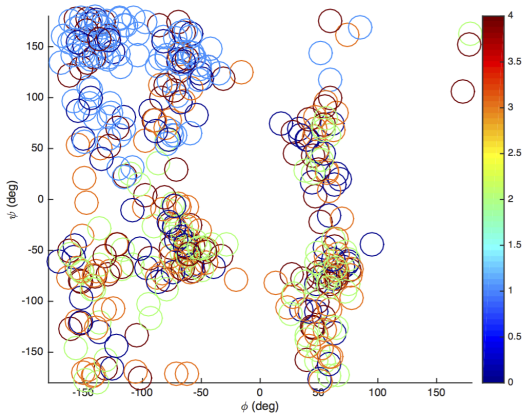

FIG. 5. Clustering results based on both the normalized second eigenvector and ball coordinate information using k-means with five clusters.

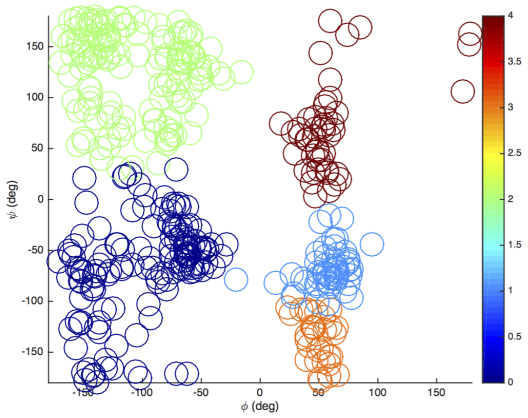

FIG. 6. Clustering results based on both the normalized second eigenvector and the first two out of six ball coordinates using k-means with five clusters.

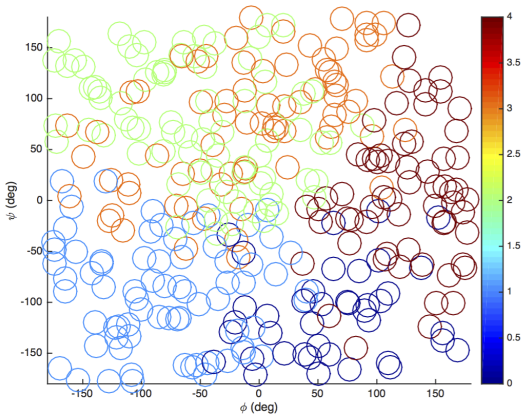

FIG. 7. Clustering results from Fig. 6 projected onto the first two principal components of the feature matrix.

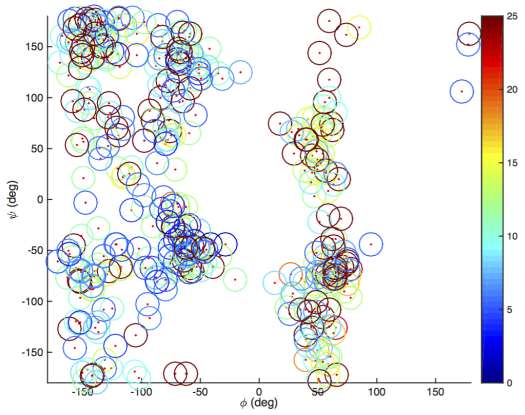

FIG. 8. Distribution of balls after clustering based on the normalized second eigenvector with three clusters, immediately **before** downsampling the number of balls.

clustering results look much more pleasing to the human eye, as seen in Fig. 6. However, clustering in fact works just as well on seven dimensions; to provide a visualization of the seven dimensions usually used for clustering, we performed PCA on the feature matrix and plotted the clustering results on the first two PCA dimensions, as seen in Fig. 7. Here, it is evident that clustering does quite a good job of separating balls that differ greatly with respect to eigenvector values and dihedral angles.

To visualize the effects of spectral clustering and subsequent downsampling of balls, we show a comparison of the number of balls before (Fig. 8) and after (Fig. 9) spectral clustering and resampling.

VI. CONCLUSION/FUTURE WORK

In summary, clustering based on the normalized second eigenvector values and ball coordinate information delivered better results than clustering based on the normalized second eigenvector values only. However, the trends regarding which cutoffs and which number of clusters to choose still remain somewhat unclear. Clustering based on the normalized second eigenvector values preserved the dynamics of the system, but often clustered states in very different dihedral angle conformations together, thus reducing the accuracy of the simulation. Because the simulation timestep was comparatively short, some of the transition matrices were noisy, resulting in outliers when clustering. Outlier detection using an Elliptic Envelope helped mitigate this problem in some cases.

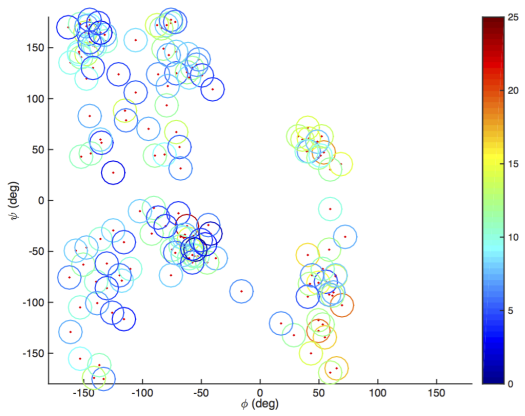

FIG. 9. Distribution of balls after clustering based on the normalized second eigenvector with three clusters, immediately **after** downsampling the number of balls.

To validate and improve the results, the main tasks would be as follows:

- Run all simulations with a timestep much larger than 40 ps, such as 100-200 ps, to obtain better estimates of the transition matrix.
- Run all simulations for a much larger number of steps. Because of limitations on the Sherlock computing cluster (`sherlock.stanford.edu`), we could run all simulations for at most 48 hours, which severely restricted the number of timesteps we could use. To calculate more meaningful statistics and evaluate the clustering results vs. non-clustering controls, running for a much larger number of timesteps is paramount.

ACKNOWLEDGMENTS

We thank Prof. Eric Darve for bringing up the topic of spectral clustering in the context of weighted ensemble. We thank our friend Ella Kim for mental support during long clustering sessions. We thank Kilian Cavalotti, who was instrumental in getting SKlearn to run on the Sherlock computing cluster.

[1]H. Feng, R. Costaouec, E. Darve, and J. Izaguirre, "A comparison of weighted ensemble and markov state model methodologies." The Journal of chemical physics **142**, 214113 (2015).
[2]T. Schlick, *Innovations in biomolecular modeling and simulations* (Royal Society of Chemistry, 2012).
[3]A. Laio and M. Parrinello, "Escaping free-energy minima," Proceedings of the National Academy of Sciences **99**, 12562–12566 (2002).
[4]D. Hamelberg, J. Mongan, and J. McCammon, "Accelerated molecular dynamics: a promising and efficient simulation method for biomolecules," The Journal of chemical physics **120**, 11919–11929 (2004).
[5]Y. Sugita and Y. Okamoto, "Replica-exchange molecular dynamics for protein folding," Chemical physical letters **314**, 141–151 (1999).
[6]G. Huber and S. Kim, "Weighted-ensemble brownian dynamics simulations for protein association reactions," Biophysical journal **70**, 97 (1996).
[7]B. Zhang, D. Jasnow, and D. Zukerman, "The weighted ensemble path sampling method is statistically exact for a broad class of stochastic processes and binning procedures," The Journal of chemical physics **132**, 054107 (2010).
[8]D. Bhatt and I. Bahar, "An adaptive weighted ensemble procedure for efficient computation of free energies and first passage rates," The Journal of chemical physics **137**, 104101 (2012).
[9]A. Dickson and C. B. III, "Wexplore: hierarchical exploration of high-dimensional spaces using the weighted ensemble algorithm." The Journal of Physical Chemistry B **118**, 3532–3542 (2014).
[10]J. Adelman and M. Grabe, "Simulating rare events using a weighted ensemble-based string method." The Journal of chemical physics **138**, 044105 (2013).
[11]G. Bowman, V. Pande, and F. Noé, *An introduction to markov state models and their application to long timescale molecular simulation* (Springer Science & Business Media, 2013).
[12]U. Luxburg, "A tutorial on spectral clustering," Statistics and Computing **17**, 395–416 (2007).
[13]A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm." Advances in neural information processing systems **2**, 849–856 (2002).