

UC Davis

UC Davis Previously Published Works

Title

Efficiently sampling conformations and pathways using the concurrent adaptive sampling (CAS) algorithm

Permalink

<https://escholarship.org/uc/item/3bm7h3pw>

Journal

The Journal of Chemical Physics, 147(7)

ISSN

0021-9606

Authors

Ahn, Surl-Hee
Grate, Jay W
Darve, Eric F

Publication Date

2017-08-21

DOI

10.1063/1.4999097

Copyright Information

This work is made available under the terms of a Creative Commons Attribution-ShareAlike License, available at <https://creativecommons.org/licenses/by-sa/4.0/>

Peer reviewed

Efficiently sampling conformations and pathways using the concurrent adaptive sampling (CAS) algorithm

Suri-Hee Ahn, Jay W. Grate, and Eric F. Darve

Citation: *The Journal of Chemical Physics* **147**, 074115 (2017); doi: 10.1063/1.4999097

View online: <http://dx.doi.org/10.1063/1.4999097>

View Table of Contents: <http://aip.scitation.org/toc/jcp/147/7>

Published by the [American Institute of Physics](#)



**COMPLETELY
REDESIGNED!**

**PHYSICS
TODAY**

Physics Today Buyer's Guide
Search with a purpose.

Efficiently sampling conformations and pathways using the concurrent adaptive sampling (CAS) algorithm

Surl-Hee Ahn,^{1,a)} Jay W. Grate,^{2,b)} and Eric F. Darve^{3,c)}

¹Chemistry Department, Stanford University, Stanford, California 94305, USA

²Pacific Northwest National Laboratory, Richland, Washington 99352, USA

³Mechanical Engineering Department, Stanford University, Stanford, California 94305, USA

(Received 8 May 2017; accepted 4 August 2017; published online 21 August 2017)

Molecular dynamics simulations are useful in obtaining thermodynamic and kinetic properties of bio-molecules, but they are limited by the time scale barrier. That is, we may not obtain properties' efficiently because we need to run microseconds or longer simulations using femtosecond time steps. To overcome this time scale barrier, we can use the weighted ensemble (WE) method, a powerful enhanced sampling method that efficiently samples thermodynamic and kinetic properties. However, the WE method requires an appropriate partitioning of phase space into discrete macrostates, which can be problematic when we have a high-dimensional collective space or when little is known *a priori* about the molecular system. Hence, we developed a new WE-based method, called the "Concurrent Adaptive Sampling (CAS) algorithm," to tackle these issues. The CAS algorithm is not constrained to use only one or two collective variables, unlike most reaction coordinate-dependent methods. Instead, it can use a large number of collective variables and adaptive macrostates to enhance the sampling in the high-dimensional space. This is especially useful for systems in which we do not know what the right reaction coordinates are, in which case we can use many collective variables to sample conformations and pathways. In addition, a clustering technique based on the committor function is used to accelerate sampling the slowest process in the molecular system. In this paper, we introduce the new method and show results from two-dimensional models and bio-molecules, specifically penta-alanine and a triazine trimer. *Published by AIP Publishing.* [<http://dx.doi.org/10.1063/1.4999097>]

I. INTRODUCTION

Computational modeling of bio-molecules is an essential tool that gives us insight into mechanisms of bio-molecules that experiments fail to capture. We can easily simulate various bio-molecules of manageable size up to microseconds, which enables us to see realistic pathways and intermediates. But it is still difficult to uncover most of the pathways and intermediates using all-atom simulations due to the temporal gap between simulations that require a time step in femtoseconds and biological systems with time scales of milliseconds. In addition, the simulated bio-molecules often stay trapped in metastable states, and no significant conformational change can be observed for a long time.

Consequently, several methods have been developed for all-atom molecular dynamics (MD) simulations to overcome these difficulties. One class of methods applies a biasing potential to force the system to move away from the metastable state, namely, umbrella sampling, metadynamics, hyperdynamics, accelerated MD (aMD),¹⁻⁸ and adaptive biasing force (ABF).⁹⁻¹⁵ Umbrella sampling can sample specific regions of phase space by adding a restraining potential to the system's potential to keep the system close to those specific regions.¹ Metadynamics, on the other hand, can quickly compute the free

energy landscape by filling the visited places with "Gaussians" or small repulsive Gaussian potentials, which forces the system to escape from local minima.²⁻⁵ Similarly, hyperdynamics and aMD can also quickly reconstruct the free energy landscape by raising the energy in low energy, metastable regions.⁶⁻⁸ Finally, ABF calculates the first derivative of the free energy landscape, which is used to bias the simulation to overcome large energy barriers and improve sampling, and ensures that the system stays close to statistical equilibrium.^{14,15}

However, these methods are only able to have a few collective variables, which is limiting in cases where many collective variables are required to characterize a bio-molecule's conformation. It is very challenging to find the few essential collective variables or reaction coordinates that can characterize the bio-molecule's conformation. In practice, it is desirable to use many of them in the hope that the true reaction coordinates are a function of the ones we selected. Additionally, metadynamics requires the collective variables to be differentiable, which is limiting in cases where we have non-differentiable, discrete collective variables, such as the number of hydrogen bonds.

Another class of methods changes the temperature of the system to sample states that are difficult to reach at the original temperature, namely, replica exchange and temperature-accelerated dynamics (TAD).^{16,17} Specifically, replica exchange enhances sampling by exchanging replicas of the same system at different temperatures periodically while maintaining an equilibrium canonical ensemble distribution

^{a)}Electronic mail: sahn1@stanford.edu

^{b)}Electronic mail: jwgrate@pnl.gov

^{c)}Electronic mail: darve@stanford.edu

for each temperature.¹⁶ Even though replica exchange maintains detailed balance for an extended ensemble of canonical states, it alters the actual kinetics of the system by exchanging states from different temperatures. Hence, we are unable to obtain real kinetic properties from replica exchange. TAD, on the other hand, raises the temperature and allows only those events that should occur at the original temperature to preserve correct dynamics.¹⁷ However, the simulated system is required to have minor anharmonic effects.

An alternate method that can be used to overcome the time scale barrier is building Markov state models (MSMs). In MSMs, the conformational space is divided into kinetically relevant macrostates and a number of trajectories are run to compute transition probabilities, overall reaction rates, and other kinetic and thermodynamic properties.^{18–22} This way, transition regions and long time scale events can be identified efficiently. In building an MSM, the lag time τ , or the time discretization of the MSM, is chosen to be long enough, and the macrostates are chosen to be small enough so that transitions are Markovian.^{19,21} However, controlling the Markovian error may be difficult or even practically impossible. One option is to use smaller macrostates such that the relaxation times inside a macrostate are very small, but this leads to a significant increase in computational cost. Another option is to increase the lag time τ . In practice though, it is very difficult to determine when the MSM is converging with respect to τ (see the work of Suárez²³ and the supplementary material). The rate is strictly determined by the eigenvalues of the infinitesimal generator of the stochastic system, and τ effectively represents a discrete approximation of this generator. As τ increases, the mean first passage time tends to increase, eventually exhibiting a linear dependency with τ . This is because as τ increases, non-Markovian effects decrease but errors due to the time discretization increase. In some cases, convergence can be very difficult to detect and may in effect never happen (see the supplementary material in Ref. 23).

To overcome the limitations that result from the Markovian assumption, we can use the weighted ensemble (WE) method instead.^{23–33} Similar to MSMs, the WE method divides the conformational space into macrostates and runs a fixed number of short trajectories or “walkers” with simulation time τ within each macrostate. The walkers carry probabilities or “weights” that sum up to 1, and these weights eventually converge to steady-state weights. Unlike MSMs, however, there is no need to adjust the simulation time τ and the macrostate decomposition to control the accuracy. This is because the Markovian assumption is not required and therefore, there is no inherent statistical bias upon convergence. In other words, the WE method yields unbiased and exact results in the absence of statistical errors (see the work of Zuckerman^{25,27} and Chap. 7 by Darve and Ryu in the work of Schlick³⁴). But because the Markovian assumption is not used, the WE method requires global convergence of the macrostate weights, which leads to a larger computational cost than MSMs (Chap. 7 in the work of Schlick³⁴). Since MSMs have uncontrollable errors unless the macrostates are chosen carefully, the WE method is preferable in many cases as it is more robust.

We now explain the WE method in more detail. To start, the collective variables to keep track of (e.g., dihedral angles

and bond distances) need to be chosen beforehand. Their values determine which macrostate each walker belongs to at each step. The macrostates form a partitioning of the collective variable space. Then the walkers are run for τ amount of time and are binned to macrostates according to their new collective variables’ values. Within each macrostate, a fixed target number of walkers are maintained by merging or splitting walkers in a statistically correct way. This process is called “resampling.” We have resampling because we want to maintain a constant stream of walkers going from one macrostate to another irrespective of the energy barrier height. If the walkers are not resampled, then the walkers would be depleted in macrostates near an energy barrier or overcrowded in macrostates at low energy and would not be able to overcome energy barriers and sample rare pathways and intermediates. Finally, these same steps are repeated until convergence.

Unfortunately, the WE method loses efficiency if the macrostates are not correctly defined. In this case, sampling relevant regions for computing reaction rates takes longer and is less accurate.³⁵ For instance, if our partitioning is fine, then we end up having too many walkers to simulate. Conversely, if our partitioning is coarse, then the walkers are unable to easily go over the energy barriers by having to go over energy barriers within the macrostate first, and trajectories of different walkers remain correlated for a long time. As a result, we end up with larger standard deviations when the macrostates are not partitioned well. Furthermore, the WE method is not ideally suited to sample high-dimensional space because of the intractable number of macrostates required to partition the space.

Considering these shortcomings of the WE method, we developed a WE-based method called the “Concurrent Adaptive Sampling (CAS) algorithm.”³² As done in the WE method, the CAS algorithm runs a number of short simulations or “walkers” for each macrostate and maintains a fixed target number of walkers for each macrostates so that macrostates are constantly sampled irrespective of their energy barriers. Unlike the original implementations of the WE method, however, the CAS algorithm constructs macrostates based on (an approximation of) the committor function, which is the probability to reach the product before the reactant from a given point. In other words, each macrostate approximates an iso-committor surface. This guarantees that the walkers can make progress in sampling the reactant-to-product pathways of interest, while keeping the computational cost under check. This idea of building macrostates that approximate the committor function has been used previously in other methods, including the string method using collective variables and optimal milestoneing.^{36,37}

Using the exact committor function would lead to an optimal partitioning of phase space and the best possible WE sampling. However, this problem in itself is as complicated as or more than computing rates and pathways of interest, so we have to use an approximation of the committor function. This can be done in different ways, and we explore two options. The first option is to compute the committor function adaptively as the simulation is on-going. We start from an initial guess and refine it using simulation data as it is produced. The second option is to run an initial brute force simulation for a short

period and approximate the committor function based on this partial sampling. An important observation is that obtaining an accurate estimation of the committor function is not required. Improvements in sampling and convergence can be observed even with a simple estimate of the committor function. In practice, we have observed that even short brute force simulations are sufficient to get a good partitioning and fast convergence of the CAS algorithm.

Besides knowing the collective variables, the reactant, and the product beforehand, we only need to know little *a priori* knowledge about the system of interest to use the CAS algorithm. That is, the CAS algorithm can be used for systems in which the reaction coordinates are largely unknown. This is because a macrostate in the CAS algorithm can be any union of n -dimensional Voronoi cells, where n denotes the number of collective variables. A Voronoi cell is a region that is defined by its center and encompasses points that are closest to the center than any other center. The construction of Voronoi cells is detailed in Sec. II B. By using n -dimensional Voronoi cells, we are able to feasibly sample high-dimensional spaces. If we let the CAS algorithm adaptively construct Voronoi cells as the simulation proceeds, then we can sample conformations and pathways without having to pre-define an intractable number of Voronoi cells in the high-dimensional space. The coupling of the CAS algorithm with any underlying MD simulation “engine” is also relatively straightforward, as done in WESTPA and Workqueue.^{29,38}

As previously mentioned, the committor function can either be computed during the simulation or before the simulation in a pre-processing step. This depends on whether we let the CAS algorithm adaptively construct macrostates as the simulation proceeds or use static macrostates throughout the simulation. If we use static macrostates, then we construct them to be isocommittor surfaces based on the (approximate) committor function. If we use adaptive macrostates, then we first let the Voronoi cells naturally follow the evolving probability distribution, which increases the number of constructed Voronoi cells and computational cost. To mitigate this problem, we propose an algorithm to down sample these Voronoi cells in an optimal way, based on the sampled data up to this point. This is done by computing the transition matrix using the existing Voronoi cells and approximating the committor function to be the second left eigenvector of the transition matrix. This is equivalent to approximating the exact committor function using piecewise constant basis functions (for which the functions are constant over each Voronoi cells). Then we define new macrostates to be a union of Voronoi cells that have the same (similar) committor function values so that each macrostate approximates an isocommittor surface. Within each macrostate, the walkers are resampled so that a fixed target number of walkers are maintained. Then we let the Voronoi cells adaptively evolve again, and this process repeats until convergence.

This way, we control the computational cost by discarding walkers that are orthogonal to the slowest pathways and keeping walkers that are progressing along the pathways. This guarantees an efficient sampling and progression along the pathways. Note that our clustering method aims to cluster Voronoi cells based on their dynamic similarity rather than

geometric or energetic similarity. Geometric clustering methods and Perron Cluster Cluster Analysis (PCCA) are sometimes used in order to build an MSM and to identify metastable states of the system, respectively.¹⁸ By clustering dynamically similar Voronoi cells via the committor function, we end up sampling dynamically important pathways and intermediates more efficiently.

This idea of using adaptive macrostates has been used previously in other methods.^{24,25,39–41} In particular, the WE-based string method adaptively constructs macrostates that form the principal reaction pathway and achieves lower error and true mean first passage times more quickly than conventional methods.⁴⁰ However, the method is poorly suited for finding multiple reactions and pathways and does not work when the reaction pathway is not well-defined. This is a common problem unless the system’s temperature is low. The minimum free energy pathway can be easily defined at zero temperature, but at higher temperatures, its definition is more ambiguous.

The WExplore method, on the other hand, adaptively constructs macrostates in a hierarchical fashion and effectively maps out the free energy landscape with pathways and intermediates that are not known *a priori*.^{41,42} To control the computational cost, the WExplore method sets a strict limit on the total number of macrostates by pre-defining the following parameters: number of hierarchy levels, macrostate sizes for each level, and maximum number of branching macrostates for each level. Although the WExplore method is able to dynamically define regions and cover the entire landscape efficiently, the critical distances associated with each level needs to be defined carefully, and since it does not use the committor function, the method is heuristic.⁴¹ In this sense, the CAS algorithm is easier and more appropriate, since we only need to define the reactant and the product and calculate the committor function to have an optimal partitioning (based on the data collected so far).

II. METHODS

A. Resampling

As done in the WE method, the walkers are resampled in the CAS algorithm after the walkers are run for τ amount of time and are binned to their corresponding macrostates according to their new collective variables’ values. That is, each walker i gets assigned to the mean weight $W = \sum_i w_i/n_w$ in its respective macrostate, and target number of walkers per macrostate or n_w walkers are maintained in each macrostate. The resampling algorithm is illustrated in Fig. 1 and was first suggested by Darve and Izaguirre (Chap. 7 in the work of Schlick³⁴). In the original WE method, walkers are assigned to weights between $W/2$ and $2W$ but not exactly equal to W .²⁴ However, it is more optimal when walkers are assigned to the same weights since variance and statistical errors are minimized (Chap. 7 in the work of Schlick³⁴). See also Ref. 25 for a description of resampling in general terms. To assign a mean weight W to each walker, we first start out with a list of walkers to process for each macrostate. Then we sort the walkers in descending order based on their weights. The sorting of the weights helps reduce data correlation, since when a walker is

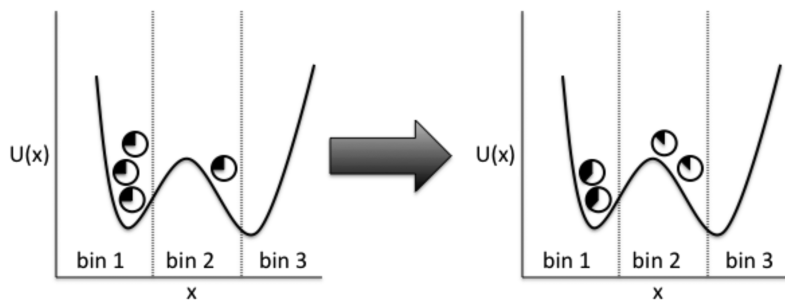


FIG. 1. Illustration of the resampling algorithm. One walker in bin 1 is killed and its probability or weight is carried by the surviving walkers. A new walker is created in bin 2 by duplicating an existing walker and dividing its weight. After resampling, each macrostate ends up with the same number of walkers and within each macrostate, the walkers end up having equal mean weight.

split, the newly created walkers become correlated for some time (Chap. 7 in the work of Schlick³⁴). Then we encounter the following two cases for the walkers:

1. $w_i \geq W$: In this case, the walker is split into an integer number of walkers of weight W , and the remainder is reinserted into the list of walkers to process.
2. $w_i < W$: In this case, the walker is merged with other walkers in a statistically exact way to create a walker with a weight $\geq W$. For instance, if we have a walker with a weight w_1 and another walker with a weight w_2 , we randomly select one of the walkers with probabilities $w_1/(w_1 + w_2)$ and $w_2/(w_1 + w_2)$, respectively, and assign the chosen walker with a weight $w_1 + w_2$ and so on.

From this procedure, we can clearly see that each walker will end up with a mean weight W , and any number of walkers can be combined or split in this fashion without violating any rules of probability. As mentioned previously, resampling is a crucial aspect of the WE method and the CAS algorithm since resampling allows rare regions to be sampled efficiently and continuously throughout the simulation.

B. Defining macrostates

As mentioned in Sec. I, the macrostates form a partitioning of the collective variable space and are essentially unions of n -dimensional Voronoi cells, where n equals the number of collective variables. We can either adaptively construct or pre-define and fix the macrostates throughout the simulation. In the case of pre-defining and fixing macrostates, Sec. II C discusses how to calculate the committor function, which is used to form the macrostates into isocommittor surfaces.

In the case of adaptively constructing macrostates, we need to use n -dimensional spheres of radius r that are used to define the newly created Voronoi cells' centers. Before the Voronoi cells are constructed, all of the walkers have run for τ amount of time and are ready to be binned to their corresponding Voronoi cells. The basic outline of how the Voronoi cells are created during a single simulation step is as follows.

1. The very first walker in the list of walkers to bin is binned to its own center. That is, a new center is created and is equal to the walker's collective variables' values.
2. For the subsequent walkers, they are tested to see if the distance between the walker's collective variables' values and any of the existing centers is less than or equal to r . If not, then a new center is created that is equal to the walker's collective variables' values.

3. If there is more than one center that is r or less away from the walker, then the walker is binned to the center that is closest to the walker.
4. After all of the walkers are binned to a center, we go through all of the walkers once more to bin them to their true closest centers, since the centers have been created for one walker at a time.
5. The centers that have no walkers are deleted, and the remaining centers become the Voronoi centers.

This adaptive construction of Voronoi cells allows us to sample unknown systems without having to partition the collective variable space beforehand, which is especially suitable for high-dimensional systems. Note that the radius r is chosen such that the walkers have relatively fast relaxation times within the resulting Voronoi cells. Therefore, r should not be too big so that there are significant energy barriers within the resulting Voronoi cells. As a rule of thumb, small Voronoi cells are suitable for systems with high energy barriers at transition regions, whereas large Voronoi cells are suitable for diffusive systems with low energy barriers.

Since the number of Voronoi cells and associated walkers can quickly increase after the simulation has proceeded for a number of steps, we need a way to control the number of walkers to make the CAS algorithm computationally tractable. The committor function is used to do that and is explained in Sec. II C.

C. Spectral clustering

As mentioned in Sec. II B, the number of Voronoi cells and associated walkers may quickly increase as the collective variable space is explored. In this case, the computational cost of running all of the walkers will be high, and sampling the slowest process of the system will take much longer. Therefore, we need to calculate the committor function to resolve this issue. The committor function describes the probability to reach the product before reaching the reactant first. If the committor function of a macrostate is 0, then the macrostate is a reactant, and if the committor function of a macrostate is 1, then the macrostate is a product. In other words, we can characterize how close the macrostate is in terms of reaching the product and exactly where the macrostate is in the reactant-to-product pathways.

In order to calculate the committor function, we first have to compute the transition matrix of the existing Voronoi cells for a number of simulation steps. This is because we use the eigenvectors of the transition matrix that are shown to be equivalent to computing an approximation of the committor

function, where the approximation is constant over each Voronoi cell, i.e., a piecewise constant approximation.⁴³ Specifically, we approximate the committor function $\psi(x)$ to be

$$\psi(x) \approx \rho_2(x)/\rho(x). \quad (1)$$

Here, $\rho_2(x)$ denotes the eigenvector corresponding to the second largest eigenvalue λ_2 , which represents the slowest process of the system. $\rho(x)$ denotes the eigenvector corresponding to the largest eigenvalue $\lambda_1 = 1$, which represents equilibrium weights of the Voronoi cells. Note that we have not normalized $\psi(x)$ so that it ranges from 0 to 1, but this is not necessary since this only shifts the values. With the committor function calculated, we can cluster the Voronoi cells by their committor function values. The union of Voronoi cells then becomes our new macrostates. Within each new macrostate, we can resample walkers and end up evenly covering the slowest pathways with walkers and reducing walkers that are redundant or orthogonal to the pathways. Thus, we naturally call this method spectral clustering. We note that our spectral clustering is slightly different from the ones in published studies, which use the first k generalized eigenvectors to cluster a high-dimensional data into k clusters using k-means.^{44,45} We only use the first two eigenvectors, and the number of clusters is pre-defined separately.

To illustrate the method more clearly, we present the basic scheme of spectral clustering in Fig. 2. Here, $\rho(x, y)$ is indicated, and the equilibrium weights are highest in the two metastable basins, which are the reactant and the product, respectively. In addition, $\rho_2(x, y)/\rho(x, y)$ is indicated, and the dynamics at this longest time scale is characterized by a global shift in probability density between the two basins, which makes sense because the equilibration between the two would take the longest than any other non-stationary process in the system. Finally, with spectral clustering, the reactant-to-product pathways are partitioned into dynamically distinct regions characterized by the values of $\rho_2(x, y)/\rho(x, y)$ and are

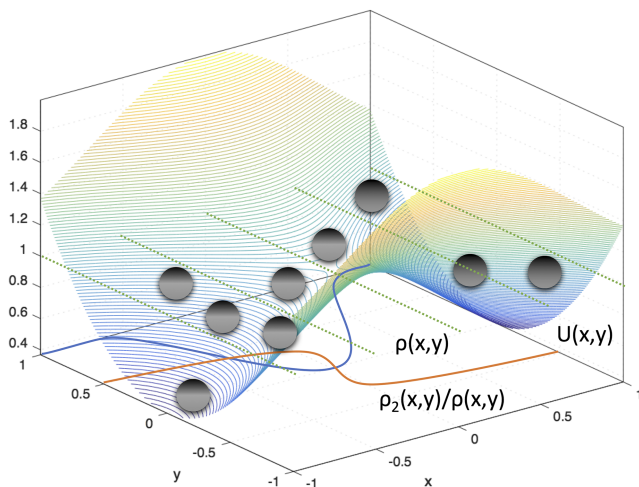


FIG. 2. Illustration of spectral clustering. Along with the free energy landscape $U(x, y)$, the equilibrium eigenvector $\rho(x, y)$ and the committor function $\rho_2(x, y)/\rho(x, y)$ are indicated. The walkers are represented as black circles, and the partitioning from spectral clustering is indicated by the dashed lines. This way, the slowest pathways, i.e., equilibration between the two metastable basins, get evenly sampled and covered while reducing walkers that are orthogonal to the pathways.

marked with dashed lines. This way, we increase sampling along the pathways while reducing sampling orthogonal to the pathways. After clustering and resampling, the walkers, which are represented as black circles, will evenly cover the pathways with new walkers in deficient regions and fewer walkers in oversampled regions.

The key idea is that we use importance sampling in spaces orthogonal to the slowest pathways. Although pathways are not represented explicitly in our method, the use of the second eigenvector allows us to use a fine discretization along the pathways, while importance sampling is used in the orthogonal directions to make sure we control the number of Voronoi cells throughout the simulation. The second eigenvector is initially computed using incomplete information. However, the information provided is sufficient to control the number of Voronoi cells constructed and make sure it remains bounded, while allowing the system to make progress along the slowest pathways. The caveat is that in all of the examples in this paper, we pre-identified and chose the reactant and the product to be the two metastable states of the system, in which case we expect the committor function and the natural reaction coordinate of the system to be nearly equivalent.⁴⁶ However, the slowest process may not always be the reaction between the two metastable states of interest as stated in Ref. 46, so caution should be exercised when using the second eigenvector to speed up the slowest process of the system.

Up to this point, the committor function has only been used to cluster the adaptively created Voronoi cells to newly define macrostates throughout the simulation. Alternatively, we can use the committor function to initially partition the collective variable space into macrostates and run the CAS algorithm with these fixed, static macrostates. The macrostates use an approximate committor function, but if we use the exact committor function, then this static partitioning can be proven to be the most optimal partitioning such that the CAS algorithm converges after one resampling step in the limit of having infinite number of walkers per macrostate or $n_w = \infty$. That is, we do not need to relax the walkers' weights to steady-state to get the correct fluxes. This choice is also optimal because the accuracy of the flux becomes independent of the relaxation times for the walkers inside each macrostate. If the macrostates are not chosen correctly, the relaxation time inside macrostates will lead to long correlation times in the flux values, resulting in larger statistical errors. With an optimal choice, the relaxation times have no effect on the standard deviation of the fluxes.

Since each macrostate has a constant committor function value, all of the walkers from a particular macrostate have the same probability of ending up in another macrostate. That is, the walkers do not need to relax or go over energy barriers within each macrostate to reach the correct flux, since they are enforced to go to their correct macrostates in the next step by being in isocommittor surfaces. Just as isocommittor surfaces are proven to be optimal milestones for milestoning, we now prove that isocommittor surfaces are optimal macrostates for the CAS algorithm.³⁷

Proposition. Let A and B be two metastable regions of interest and Ω denote the entire collective variable space. The following assumptions are made:

1. The system's dynamics obeys detailed balance or time reversibility.
2. Macrostates are constructed such that the committor function value is constant in each macrostate, i.e., they are isocommittor surfaces.
3. $n_w = \infty$.

The result is then all of the walkers in each macrostate, regardless of the walkers' distribution and positions, have the same probability to end up in another macrostate. Said otherwise, it is not necessary to relax the distribution of walkers inside each macrostate to obtain the correct fluxes between A and B and vice versa. Hence, we obtain exact fluxes going from A to B and vice versa in one step.

Proof. First, let $q(x)$ denote the forward committor function or the solution of the following Dirichlet partial differential equation problem with respect to L , which denotes the infinitesimal generator of the diffusion that governs the system's dynamics,

$$(Lq)(x) = 0 \text{ if } x \in \Omega \setminus (A \cup B),$$

$$q(x) = 0 \text{ if } x \in A, \quad (2)$$

$$q(x) = 1 \text{ if } x \in B.$$

The solution of Eq. (2) is the committor function that describes the probability to reach product B before reactant A . Additionally, $1 - q(x)$ denotes the backward committor function.

Now, let N be the number of macrostates that divide up $q(x)$ and $z_1 = 0 < z_2 < z_3 < \dots < z_N = 1$ denote uniformly spaced committor function values. Also, let m_{ij} , $j = 1, \dots, n_i$ denote a Voronoi cell that has a center x_{ij} and a committor

function value of z_i , and n_i be the total number of those Voronoi cells. Then we can construct the macrostates to be the following isocommittor surfaces or union of Voronoi cells that have the same committor function value:

$$M_i = \bigcup_{j=1}^{n_i} m_{ij} = \bigcup_{j=1}^{n_i} \{x \mid |x - x_{ij}| \leq |x - x_{kl}|, ij \neq kl, \quad (3)$$

$$q(x) = z_i\}, i = 1, \dots, N.$$

With this setup, A can be re-labeled as M_1 and B as M_N . From Eq. (3), we can expect the macrostates close to M_1 to have $0 < q(x) \ll 1$ and the macrostates close to M_N to have $0 \ll q(x) < 1$. Now, we will prove that all of the walkers in M_i have the same probability to end up in another macrostate M_j , regardless of where the walkers are located within M_i . Note that this is typically not the case because where the walker ends up in the next step depends on its position within the current macrostate.

To see this, let the reactant be the union of all of the states left of M_i , i.e., $\tilde{M}_1 = \bigcup_{i=1}^{i-1} \bigcup_{j=1}^{n_i} m_{ij}$, a union of macrostates that have a committor function value $\leq z_{i-1}$, and the product be the union of all of the states right of M_i , i.e., $\tilde{M}_N = \bigcup_{i=i+1}^N \bigcup_{j=1}^{n_i} m_{ij}$, a union of macrostates that have a committor function value $\geq z_{i+1}$, and consider this new reaction. Then M_i becomes an isocommittor surface for this reaction as well, and the committor function for this reaction becomes $\tilde{q}(x) = (q(x) - z_{i-1}) / (z_{i+1} - z_{i-1}) = (z_i - z_{i-1}) / (z_{i+1} - z_{i-1})$. Indeed, $\tilde{q}(x)$ still satisfies Eq. (2) and boundary conditions, since $\tilde{q}(x) = 0$ if $x \in \tilde{M}_1$ and $\tilde{q}(x) = 1$ if $x \in \tilde{M}_N$. Returning to our original setup, we get the following probabilities p_{ik} to go to M_k after being in M_i ,

$$p_{ik} = \begin{cases} \frac{z_i - z_{i-1}}{z_{i+1} - z_{i-1}}, & \text{if } k = i + 1, i = 2, \dots, N - 1, \\ 1 - \frac{z_i - z_{i-1}}{z_{i+1} - z_{i-1}} = \frac{z_{i+1} - z_i}{z_{i+1} - z_{i-1}}, & \text{if } k = i - 1, i = 2, \dots, N - 1, \\ 1, & \text{if } i = 1, k = 2 \text{ or } i = N, k = N - 1, \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

Taken together, all of the walkers in M_i reach M_k with the same probability p_{ik} , regardless of the walkers' distribution, since each M_i is an isocommittor surface. Since no systematic errors are present and statistical errors become zero in the limit of $n_w = \infty$, we obtain the exact fluxes going from A to B and vice versa in one step in the limit of $n_w = \infty$. \square

Now that we have proven that using the committor function to create macrostates is the optimal choice, the specific steps taken for spectral clustering are listed below.

1. After the simulation has a number of Voronoi cells that is equal to or greater than the pre-defined threshold number of Voronoi cells, a transition matrix T of existing Voronoi cells is calculated for a number of simulation steps. The Voronoi cells are fixed during these simulation steps so that T can be calculated. Each entry T_{ij} represents the weight that transitioned from the Voronoi

cell i to Voronoi cell j in one step. This can be done since each walker keeps track of its weight, previous coordinates, and current coordinates. However, since a transition matrix needs to fulfill detailed balance when a canonical ensemble simulation is run under Hamiltonian dynamics, each entry T_{ij} is computed using the equations in Ref. 47, which are

$$C_{ij} = \frac{B_{ij} + B_{ji}}{2s} = C_{ji}, \quad (5)$$

$$T_{ij} = \frac{C_{ij}}{\sum_{k=1}^n C_{ik}}. \quad (6)$$

Here, B_{ij} denotes the sum of the weights that went from the Voronoi cell i to Voronoi cell j during s number of steps, and C_{ij} denotes the state-to-state time-correlation estimator that fulfills detailed balance. Using C_{ij} , we

can calculate T_{ij} using Eq. (6). The detailed balance requirement also reduces the uncertainties of the committor function.⁴⁸ This transition matrix T corresponds to the “graph Laplacian matrix” L described in Refs. 44 and 45.

2. Then an eigendecomposition on T is performed. From the properties of a transition matrix, the eigenvector corresponding to the largest eigenvalue $\lambda_1 = 1$ or $\rho(x)$ represents the equilibrium eigenvector, as previously mentioned. The rest of the eigenvectors correspond to non-stationary processes that are slower for eigenvectors corresponding to eigenvalues close to 1.
3. Using the equilibrium eigenvector, the eigenvector corresponding to the second largest eigenvalue λ_2 or $\rho_2(x)$ is divided by $\rho(x)$ to be a good approximation of the committor function, as previously stated in Eq. (1). The $\rho_2(x)/\rho(x)$ entries represent probabilities of going from the reactant to product and are the right quantities to use for clustering, since we will increase sampling along the reactant-to-product pathways while preserving the dynamics of the system. Hence, $\rho_2(x)/\rho(x)$ is used to cluster the Voronoi cells using k-means, and the resulting union of Voronoi cells become the new macrostates. The number of clusters is set beforehand.
4. In resampling the new macrostates, the target number of walkers per cluster or macrostate is set so that the total number of walkers will be reduced overall. After resampling, the most probable Voronoi cells end up being mostly populated within the new macrostates.

After spectral clustering is performed as listed above, normal CAS algorithm steps are taken until the number of Voronoi cells hits a certain threshold, which signals spectral clustering to be performed again. This method is illustrated more clearly in Sec. III, where specific examples are discussed.

III. EXAMPLES

A. Two-dimensional model

For the simplest case, the CAS algorithm is tested on a two-dimensional potential surface described by $U(x, y) = e^{-x^2} + y^2$ taken from Ref. 34. The potential surface is bounded by $-1.0 \leq x \leq 1.0$ and $-1.0 \leq y \leq 1.0$. The Metropolis algorithm is used to move the walkers with $\Delta x = 0.05$ or $\Delta y = 0.05$ once per simulation step, and the target number of walkers per macrostate n_w is set to 100 for all of the simulations. As seen in Fig. 3, the metastable states are $A = \{(x, y) : -1.0 \leq x \leq 0.0, -1.0 \leq y \leq 1.0, \sqrt{(x+1)^2 + y^2} \leq 0.4\}$ and $B = \{(x, y) : 0.0 \leq x \leq 1.0, -1.0 \leq y \leq 1.0, \sqrt{(x-1)^2 + y^2} \leq 0.4\}$, and the minimum energy pathway is a straight pathway linking these two states. The inverse temperature β is set to 10.0, and half of the walkers initially start from $(-0.6, 0.0)$, or A , and the other half start from $(0.6, 0.0)$, or B . Since the initialization of walkers impacts the convergence time of the fluxes, the initial conditions are picked such that the simulations will reach convergence quickly. The forward (from A to B) and backward (from B to A) fluxes are measured by labeling walkers from A and B with colors, changing colors when walkers from A

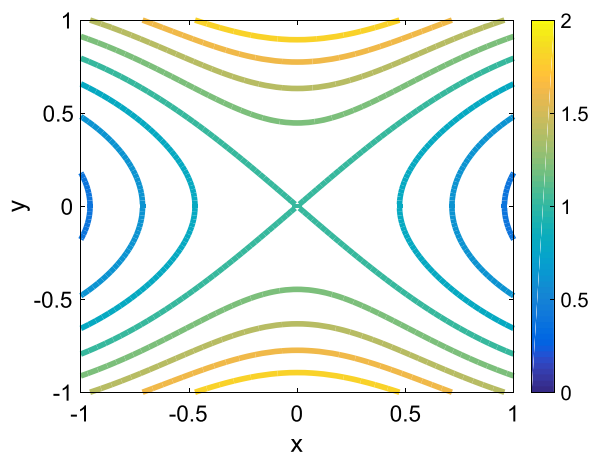


FIG. 3. Two-dimensional potential surface described by $U(x, y) = e^{-x^2} + y^2$. The color bar indicates energy values.

reach B and vice versa, and resampling each color within each Voronoi cell, as done in Ref. 34.

All of the simulations have the same fixed running time of 48 h to ensure that the computational cost is the same across all simulations. The simulations differ from each other by having different radii r and/or having spectral clustering turned on or not. These different simulation conditions are chosen to demonstrate the power of the CAS algorithm with spectral clustering compared with “conventional” methods without spectral clustering. The committor function is used for spectral clustering to efficiently sample the slowest process, which in this case is the equilibration between the two metastable states A and B . Three simulation runs are done for each kind of simulation, and the standard deviation of the three runs is multiplied by 2, which approximately represent 95% confidence interval, for error bars. The total simulation time is calculated by the cumulative total number of macrostates \times number of walkers per macrostate.

To check whether the CAS algorithm with spectral clustering is more efficient than “conventional” methods without spectral clustering, we plotted and compared the forward and backward fluxes between A and B . Indeed as seen in Fig. 4, the CAS algorithm with spectral clustering is the one that is closest to converging to the correct fluxes with smaller error bars within the fixed running time. When the Voronoi cells are chosen to be big ($r = 1.0$), we are able to obtain the correct fluxes but the error bars are bigger by having to go over energy barriers and taking time to reach relaxation within the larger macrostates. In contrast, when the Voronoi cells are chosen to be small ($r = 0.4$), the number of Voronoi cells grows rapidly, and we end up wasting our efforts in covering every region of the potential surface. Since the potential surface is bounded, the number of Voronoi cells does not grow beyond a certain point. Nonetheless, the computational cost becomes large when many Voronoi cells are created, and spectral clustering proves to be useful in this scenario. With spectral clustering, we are able to focus our efforts in having walkers equilibrate between A and B and sample this slowest process and the intermediates involved.

To see how spectral clustering works in this example, we plotted simulation snapshots in Fig. 5. Spectral clustering or

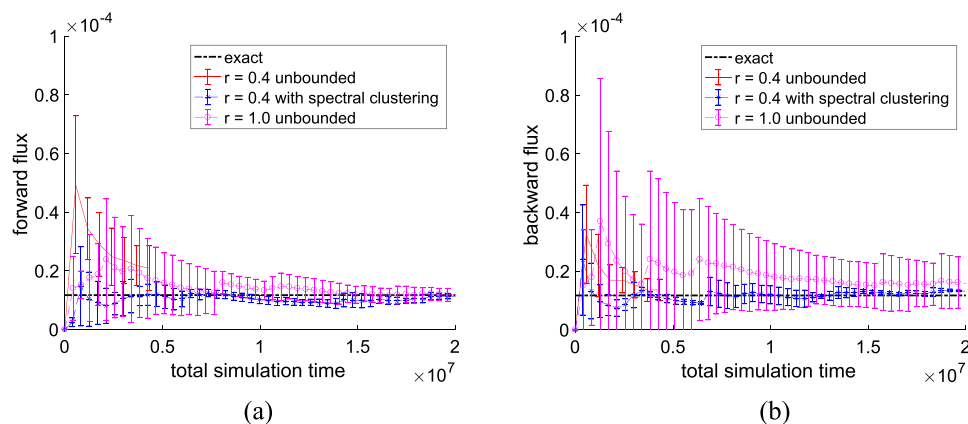


FIG. 4. Flux comparisons among the CAS algorithm simulations with and without spectral clustering. In the case of $r = 0.4$ unbounded simulations, the number of macrostates grows rapidly, and we end up wasting our efforts in covering every region. With spectral clustering, we are able to focus our efforts in having walkers rapidly equilibrate between the reactant and product. For these particular spectral clustering simulations, spectral clustering starts when the number of Voronoi cells is 8 or larger, the number of cluster is set to 5, the total number of walkers per cluster is set to 80, and the transition matrix is calculated for 5 steps. (a) Forward flux (A to B). (b) Backward flux (A to B).

calculation of the transition matrix starts from step 19 when the total number of Voronoi cells becomes large. After calculation of the transition matrix is finished at step 69, the committor function is calculated to cluster the Voronoi cells, and the unions of Voronoi cells become the new macrostates. We can see that the macrostates at the very left and right-hand sides are the largest, which makes sense since most of the walkers are concentrated near the minimum energy basins A and B and have small weight changes, making them relatively stable. After resampling, however, each macrostate ends up with the same number of walkers, which results in spreading walkers around minimum energy pathway and reducing walkers orthogonal to the pathway. Hence, we can see that the CAS algorithm with spectral clustering is very effective at focusing efforts on sampling slow time scales and is much more efficient than conventional methods, even for this relatively simple two-dimensional model example.

B. Penta-alanine

After the low-dimensional model was tested, we applied the CAS algorithm to high-dimensional real examples. Namely, we applied our method to penta-alanine, which consists of five alanine residues and 66 atoms. Its conformations can be described by the three middle ϕ and ψ dihedral angle pairs or six dihedral angles in total.⁴⁹ If all three residues are helical α_R states ($-100.0^\circ \leq \phi \leq -30.0^\circ$ and $-90.0^\circ \leq \psi \leq -10.0^\circ$), then penta-alanine is considered to be folded, whereas if all three residues are coiled C_{7eq} states ($-180.0^\circ \leq \phi \leq -55.0^\circ$ and $105.0^\circ \leq \psi \leq 180.0^\circ$ or $-180.0^\circ \leq \psi \leq -155.0^\circ$), then penta-alanine is considered to be unfolded. Since its conformations have to be described in high dimensions, penta-alanine is a nontrivial example. The fluxes between the two states, or the folding and unfolding fluxes, are of interest to us. Again, these fluxes are measured by labeling walkers from folded and unfolded states with colors, changing colors when walkers from one state reach another, and resampling each color within each macrostate, as done in Sec. III A.

The MD simulations are run with Gromacs 4.5.7 using Amber96 force field and implicit solvent at temperature $T = 300$ K with time step $\Delta t = 2$ fs.^{50,51} For brute force simulations, five different initial configurations (folded, unfolded, partially folded, partially unfolded, and neither folded nor unfolded) are used, and each simulation is $3 \mu\text{s}$ long. The standard deviation of the five runs is multiplied by 2, which approximately represent 95% confidence interval, for error bars. The unfolding and folding fluxes from brute force simulations are listed in Table I. For the CAS algorithm simulations, the same MD simulation conditions are used and since all of the collective variables are dihedral angles with $[-180^\circ, 180^\circ]$ limits, the minimum distance is taken between the previous and the new dihedral angle values when creating macrostates for the walkers.

First, we evaluated the effectiveness of the CAS algorithm compared to conventional brute force simulations. In order to do so, we calculated and plotted the unfolding and folding fluxes to compare. We use the bootstrapping procedure, or drawing first passage times randomly with replacement for a number of times that is proportional to the total simulation time, to obtain the fluxes and to simulate the statistics which we would obtain with shorter simulations. For comparison, the CAS algorithm simulations are run with simulation time $\tau = 500.0$ ps, which happens to fulfill the Markovian property according to Ref. 49, even though it is not necessary for the CAS algorithm. Since the fluxes are not very low, i.e., transitions occur repeatedly due to having low energy barriers, the macrostates are pre-defined and fixed throughout the simulation in this case. The folded and unfolded states are defined as single states, and the rest are partitioned using the committor function, which prove to give the most optimal macrostates.

Specifically, a single $3 \mu\text{s}$ brute force simulation trajectory is used to sample the collective variable space, which is then covered with Voronoi cells. These Voronoi cells are used to calculate the transition matrix so that we can obtain the committor function. States are sampled every 500.0 ps so that the transition matrix is Markovian, and r is set to 80.0

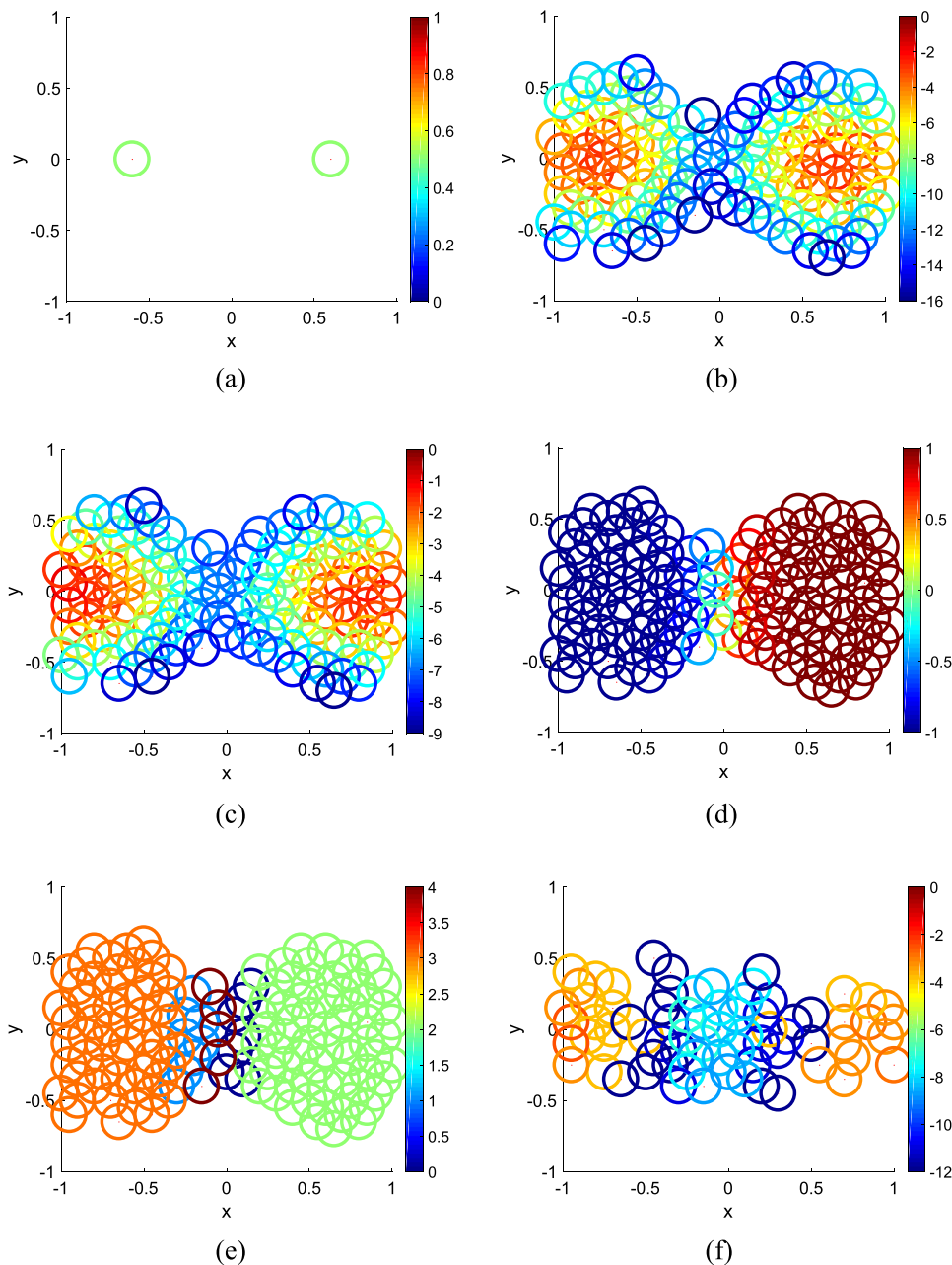


FIG. 5. Spectral clustering simulation snapshots for $r = 0.1$ at $\beta = 10.0$. The simulation starts with half of the walkers on the left metastable state and the other half of the walkers on the right metastable state, as shown in (a). Voronoi cells are colored according to their weights in log scale in (b), (c), and (f), and the color bar indicates which colors correspond to which weight values in log scale. In (d), the macrostates are colored according to their committor function values. In (e), each macrostate, a union of Voronoi cells, is represented by a color (five clusters in total, numbered from 0 to 4). For this particular simulation, spectral clustering starts when the number of Voronoi cells is 120 or larger, the total number of clusters is set to 5, the total number of walkers per cluster is set to 50, and the transition matrix is calculated for 50 steps. (a) Initial condition. Step 0. (b) Transition matrix calculation starts. Step 19. (c) Equilibrium weights. Step 69. (d) Committor function. Step 69. (e) Clusters. Step 69. (f) After clustering and resampling. Step 69.

so that approximately 200 Voronoi cells are used to cover all of the states, a size that is shown to give an accurate Markov State Model (MSM) or transition matrix according to Ref. 49. Similar to spectral clustering, the committor function is used to cluster neither the folded nor unfolded or intermediate states into macrostates with approximately constant committor function values. In particular, the minimum and the maximum committor function values are obtained and are used to create an interval of committor function values. This interval is evenly divided into a pre-defined number of clusters or macrostates,

TABLE I. Penta-alanine fluxes from brute force simulations at $T = 300$ K.

	Unfolding (folded to unfolded)	Folding (unfolded to folded)
Flux (ns^{-1})	0.0320 ± 0.0035	0.0473 ± 0.0053
Error (%)	10.79	11.24

and the intermediate states are binned to their corresponding macrostates according to their committor function values. This way, every part of the committor function can be efficiently sampled throughout the simulation. The pre-defined number of macrostates is set to be large enough so that none of the macrostates will be empty and to have the macrostates represent a narrow range of committor function values so that they are close to being true isocommittor surfaces. In this case, the number of macrostates is chosen to be 10, which excludes the folded and unfolded states. As a result, we end up with a single folded state, a single unfolded state, and 10 states that represent probabilities going from folded to unfolded. With these macrostates combined with binning and resampling, the walkers are forced to make progress along the folding/unfolding transition pathways as the simulation proceeds.

With this pre-defined and fixed macrostate setup, we only need to initialize walkers and choose n_w to run the most optimal

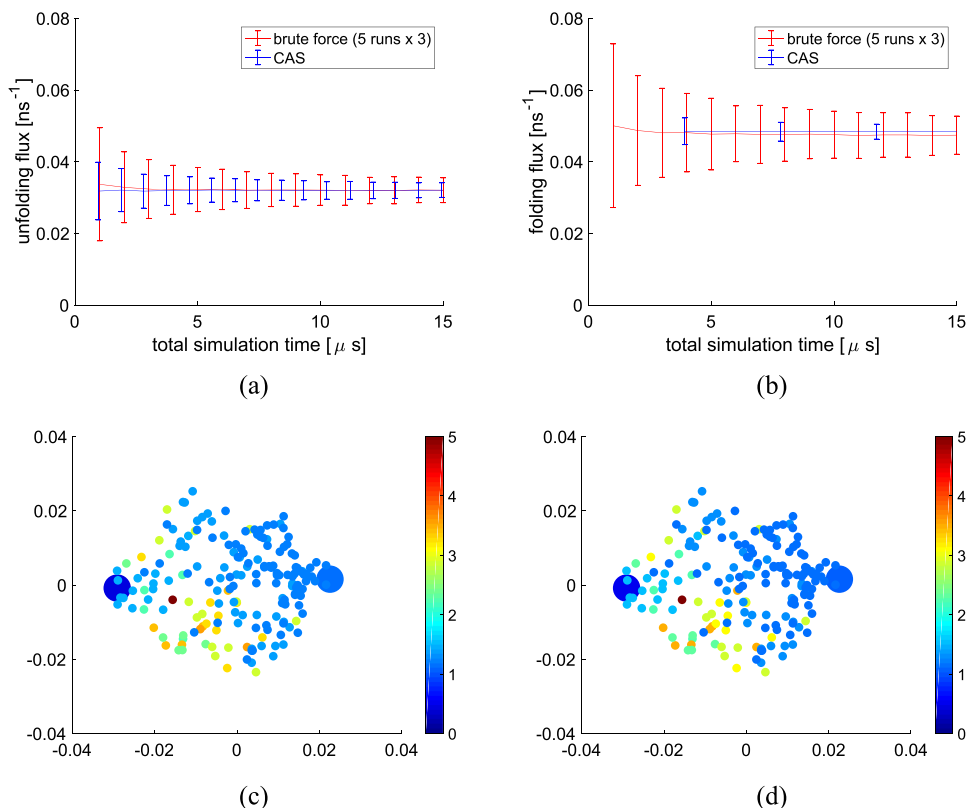


FIG. 6. Flux and free energy landscape comparisons between brute force and the CAS algorithm simulations at $T = 300$ K. (a) and (b) show that the CAS algorithm simulations are able to obtain the correct fluxes while having significantly smaller error bars. (c) and (d) show that the CAS algorithm simulations are also able to reproduce the correct free energy landscape, and the macrostates are colored according to their free energy values or $-k_B T \ln P$, where P equals the weight. The larger circular macrostates on the left-hand side and the right-hand side represent the folded state and the unfolded state, respectively. ϵ for diffusion map, which is used for visualization purposes only, is set to 10, and the number of macrostates (excluding the folded and unfolded states) is set to 10. (a) Unfolding flux. (b) Folding flux. (c) Free energy landscape from brute force. (d) Free energy landscape from CAS.

CAS algorithm simulations. First, set n_w such that it represents numbers of walkers per macrostate per “color.” In this case, the “color” represents whether the walker comes from the folded or unfolded state. For instance, if a macrostate has walkers from both the folded and unfolded states, then after resampling, the macrostate will end up with n_w walkers that come from the folded state and n_w walkers that come from the unfolded state. Empirically, setting n_w to be the average number of walkers that initially come from the folded state is found to yield the most accurate unfolding flux and vice versa for the folding flux. Hence, n_w is chosen to be 90 for the unfolding flux and 230 for the folding flux from using the given brute force simulation trajectory. Finally to initialize walkers, all of the states need to make a transition from their initial states to their next states according to what their next states are from the brute force trajectory and resampled according to the pre-defined n_w as previously stated. The resulting walkers are used as initial walkers for the CAS algorithm simulations.

To directly compare the CAS algorithm’s accuracy and efficiency with brute force simulations, the CAS algorithm

TABLE II. Penta-alanine fluxes from the CAS algorithm simulations at $T = 300$ K.

	Unfolding (folded to unfolded)	Folding (unfolded to folded)
Flux (ns^{-1})	0.0320 ± 0.0020	0.0484 ± 0.0018
Error (%)	6.15	3.72
Reduction in error $\left(\frac{\text{brute force error}}{\text{CAS error}} \right)$	1.75	3.02

simulations are run for $15 \mu\text{s}$, which is equal to the cumulative total number of walkers \times simulation time ($\tau = 500.0$ ps). As seen in Fig. 6 and in Table II, the performance of the CAS algorithm is significantly better than brute force in getting the correct fluxes efficiently with much smaller error bars.

Second, we plotted the free energy landscapes obtained from brute force and the CAS algorithm simulations to further validate the CAS algorithm’s accuracy. To visualize a high-dimensional bio-molecule such as penta-alanine, we used diffusion map to project the high-dimensional space onto two dimensions.^{52,53} Diffusion map is a non-linear dimensionality reduction technique that discovers the underlying low-dimensional manifold, preserves the true geometric structure, and is robust to noise perturbation.⁵³ The parameter ϵ for diffusion map corresponds to the neighbor size and is chosen so that the underlying manifold is clearly shown and not entirely uniformly distributed from one another. For both brute force and the CAS algorithm simulations, $15 \mu\text{s}$ of simulation data is used to plot the free energy landscape. As mentioned previously, the folded and unfolded states are fixed to be single states and the rest are clustered according to their committor function values. Thus, as seen in Fig. 6, the folded state (left) and the unfolded state (right) are represented as larger circular macrostates, and the rest are clustered and colored according to the macrostates’ weights. The free energy landscapes

TABLE III. Penta-alanine fluxes from brute force simulations at $T = 250$ K.

	Unfolding (folded to unfolded)	Folding (unfolded to folded)
Flux (ns^{-1})	0.00198 ± 0.000739	0.0311 ± 0.0116
Error (%)	37.36	37.27

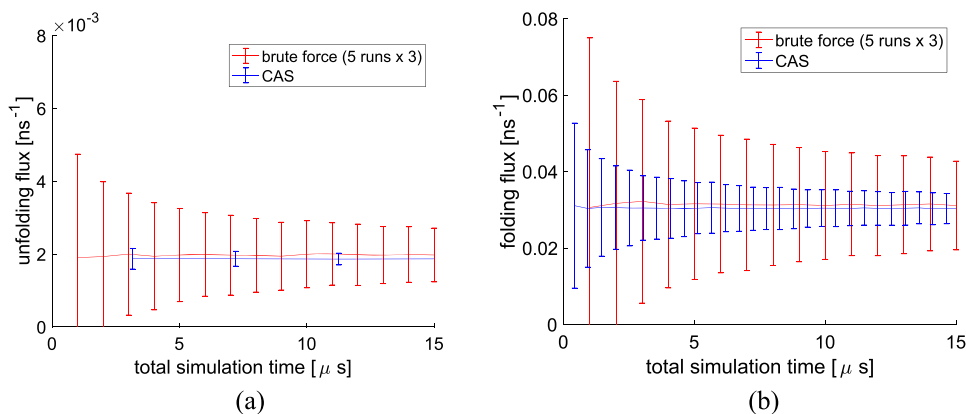


FIG. 7. Flux comparisons between brute force and the CAS algorithm simulations at $T = 250$ K. (a) Unfolding flux. (b) Folding flux.

from brute force and the CAS algorithm simulations are almost identical, which validates the sampling accuracy of the CAS algorithm.

To see more of CAS algorithm's efficiency over brute force simulations, we lowered the system temperature from $T = 300$ K to $T = 250$ K, which made the transitions significantly rarer. Again, brute force simulations are run for $15 \mu\text{s}$ to calculate the brute force fluxes, and a single $3 \mu\text{s}$ brute force simulation trajectory is used to partition the free energy landscape beforehand with the committor function for the CAS algorithm. The unfolding and folding fluxes from brute force simulations are listed in Table III. For the CAS algorithm simulations at $T = 250$ K, n_w is chosen to be 150 for the unfolding flux and 20 for the folding flux from using the given brute force simulation trajectory. As seen in Fig. 7 and Table IV, the reduction in error is much greater at lower temperatures.

Finally, we extracted the major conformations for each macrostate to check whether they correspond to the correct intermediate states according to their ranges of committor function values. As expected, the conformations' degree of foldedness/unfoldedness matched with their committor function values and the folded states (labeled as FFF) gradually unfolded one by one as they got nearer to the unfolded states (labeled as UUU) and vice versa, as seen in Fig. 8. Interestingly, none of the intermediate macrostates between folded and unfolded states, except for the one nearest the unfolded state, has the first and the second ϕ, ψ pairs unfolded. This is consistent with the claim that the first ϕ, ψ pair, which is nearest to the N terminus, has the slowest relaxation to unfold, as stated in Ref. 54. Accordingly, penta-alanine most likely unfolds like a zipper starting from the C terminus and ending with the N terminus.

TABLE IV. Penta-alanine fluxes from the CAS algorithm simulations at $T = 250$ K.

	Unfolding (folded to unfolded)	Folding (unfolded to folded)
Flux (ns^{-1})	0.00187 ± 0.000141	0.0303 ± 0.0040
Error (%)	7.52	13.17
Reduction in error $\left(\frac{\text{brute force error}}{\text{CAS error}} \right)$	4.97	2.83

Taken together, the CAS algorithm is not only able to efficiently obtain kinetic pathways and fluxes for penta-alanine, but is also able to extract useful thermodynamic information like transition states and free energy landscapes. With the use of diffusion map, we are also able to visualize the high-dimensional conformational space and pathways going from one state to another.

C. Triazine trimer

For the final example, the CAS algorithm is applied to a high information content triazine polymer newly developed by Grate and co-workers at Pacific Northwest National Laboratory.⁵⁵ The triazine polymers encode information by having various side chains and since they do not have hydrolyzable bonds, the molecules are robust and are not susceptible to proteases.⁵⁵ Although the triazine polymers have been shown to form particular sequential stacks, have stable backbone-backbone interactions through hydrogen bonding and pi-pi interactions, and conserve their *cis/trans* conformations throughout the simulation, there are still many questions left to be solved. We do not know their various possible conformations along with their probabilities of occurring and their rare pathways and probabilities of *cis-to-trans* transitions.

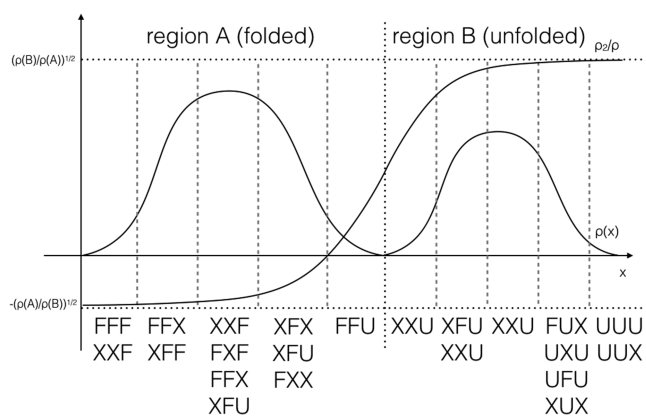


FIG. 8. Diagram of the committor function and macrostates for penta-alanine at $T = 300$ K. $\rho(x)$ indicates the equilibrium weights and ρ_2/ρ indicates the committor function. The committor function is uniformly divided into 10 macrostates. The folded state (labeled as FFF) and the unfolded state (labeled as UUU) have committor function values that are within the committor values of the leftmost and rightmost macrostates, respectively, so they are marked as the major conformations here but are separate macrostates from the leftmost and rightmost macrostates.

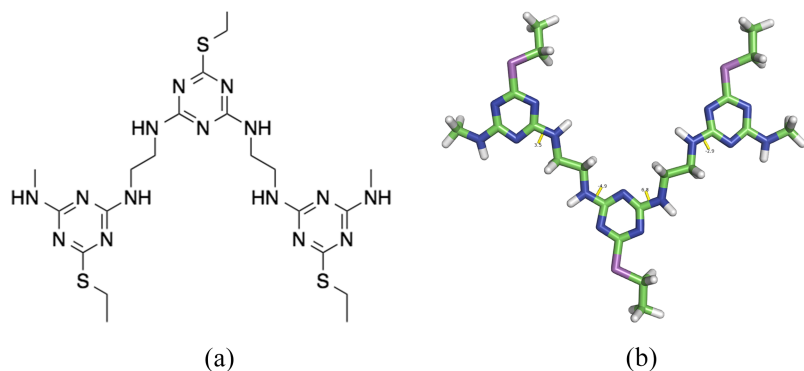


FIG. 9. Structure of all-*cis* triazine trimer. (a) Molecular structure. (b) Pymol structure with *cis/trans* dihedral angles marked.

Figure 9 shows the structure of a triazine trimer where all the aromatic ring to alpha-nitrogen bonds in the main chain are in the *cis* configuration, i.e., all-*cis* or CCCC, where C denotes *cis*, since there are four *cis* bonds.

To demonstrate that the CAS algorithm can capture rare pathways not easily accessible by regular MD simulations, the CAS algorithm is used to sample *cis*-to-*trans* transitions. For this, a single all-*cis* triazine trimer (CCCC) is simulated with

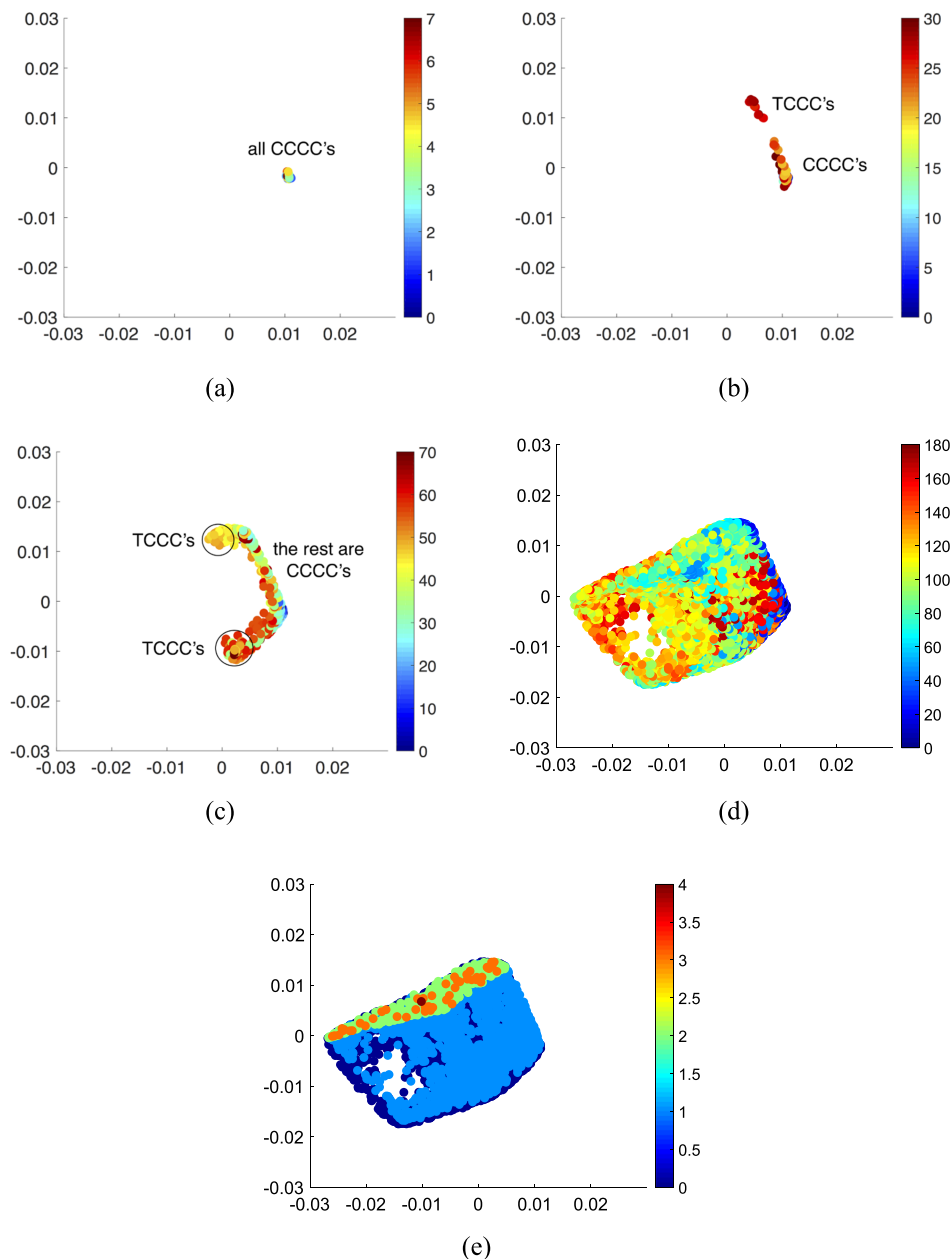


FIG. 10. Simulation snapshots of all-*cis* triazine trimer (CCCC) transitioning to all-*trans* triazine trimer (TTTT), where C denotes *cis* and T denotes *trans*. Note that the C's and T's are written in order, so TCCC refers to a triazine trimer with one *trans* bond at the end of the triazine trimer. The Voronoi cells are colored according to their free energy values or $-k_B T \ln P$, where P equals the weight, except for (e) where the color bar indicates the number of *cis* to *trans* transitions. ϵ for diffusion map, which is used for visualization purposes only, is set to 1. (a) Step 50. (b) Step 100. (c) Step 200. (d) Step 400. (e) Number of *cis* to *trans* transitions.

Gromacs 4.6.4 using implicit solvent at temperature $T = 300$ K with time step $\Delta t = 2$ fs.⁵⁰ Most simulation parameters are identical to the ones in Ref. 55, including the force field that was generated using the generalized Amber force field (GAFF) and is used with a Generalized Born/Surface Area (GBSA) implicit solvation model.^{51,56} Otherwise, the radius r is set to 24° , the target number of walkers per macrostate n_w is set to 10, and the simulation time τ is set to 0.04 ps. The collective variables are the four dihedral angles that are associated with bonds that can be *cis* or *trans*, marked as all *cis* in Fig. 9. Note that these are not exactly the same as the conventional ω dihedral angles, which determine the *cis/trans* conformation in peptide bonds. But like the regular ω dihedral angles, the molecule is *cis* when the dihedral angles are all equal to 0° and *trans* when they are all equal to 180° . Again as in Sec. III B, since all of the collective variables are dihedral angles with $[-180^\circ, 180^\circ]$ limits, the minimum distance is taken between the previous and the new dihedral angle values when creating Voronoi cells for the walkers.

Since the energy barrier to get to the *trans* conformation is very high (a single transition is experimentally measured to have an energy barrier of $\Delta G^\ddagger = 15$ kcal/mol), the walkers tend to cluster around the *cis* region initially.⁵⁷⁻⁵⁹ However, with a very short resampling time, small Voronoi cells, and enough number of walkers per macrostate, the all-*cis* triazine trimer (CCCC) is able to go from *cis* to *trans* one at a time and eventually, it transitions into an all-*trans* triazine trimer (TTTT, where T denotes *trans*) as seen in Fig. 10. Specifically, we start to see triazine trimers with one *trans* bond at the end of the triazine trimers (TCCC, where C's and T's are written in order) at step 100, and we see more TCCC's at step 200. At step 400, most of the free energy landscape gets covered, and we see triazine trimers with more than one *trans* bond, in particular, the first appearance of an all-*trans* triazine trimer as indicated by the red dot (representing four *cis* to *trans* transitions) in Fig. 10(e). Again since this is a four-dimensional problem, diffusion maps are used to visualize each step of the CAS algorithm simulation. Note that at step 400, the CAS simulation has not converged yet, since all-*trans* triazine trimer has an infinitesimally small weight. But we expect the all-*trans* triazine trimer to have a comparable weight to that of all-*cis* triazine trimer since both conserve their *cis/trans* conformation throughout long MD simulations (data not shown).

From running the CAS algorithm simulation, we were able to observe not just one but four *cis*-to-*trans* transitions, whereas observing even one *cis*-to-*trans* transition in conventional MD simulations is not feasible due to its high energy barrier. With the CAS algorithm, rare *cis*-to-*trans* transitions were easily seen with optimal choice of parameters, and indeed, all four *cis*-to-*trans* transitions could be observed.

IV. DISCUSSION

The CAS algorithm is an efficient enhanced sampling method that combines the WE method to overcome energy barriers and a method based on the transition matrix to reduce redundant walkers that do not allow the simulation to efficiently make progress along the slowest reaction. The

important features of the WE method that are incorporated into the CAS algorithm are the following.

1. There is no Markovian error as in MSMs, which requires a global convergence of the weights. This is a necessary trade-off, however, resulting from the fact that Markovian approximation is not used and therefore, we need to wait until steady-state is reached, at which point the walkers are correctly distributed and non-Markovian effects disappear (Chap. 7 in the work of Schlick³⁴).
2. Exact fluxes are obtained upon convergence with no bias.
3. General collective variables, including non-differential variables such as discrete coordinates with integer or Boolean values, can be considered. This will be useful for further studying the triazine polymers' self-assembly, e.g., have the number of hydrogen bonds and pi-pi interactions as a collective variable.
4. The MD simulation program can be used in a black box manner, i.e., the wrapper Python CAS algorithm code can be used with any MD simulation program without having to modify its source code and is available at http://github.com/shirleyahn/CAS_Code. A complete documentation on how to make use of the CAS algorithm code is still underway.
5. There is a large amount of parallelism in the algorithm, since we simultaneously run many walkers for each macrostate. This allows us to achieve computational efficiency proportional to the available computational resources.

The novel features of the CAS algorithm are the following.

1. Mild assumptions or little *a priori* knowledge about the system is required, since we can use many collective variables for systems in which the reaction coordinates are unknown. The macrostates are high-dimensional and if we adaptively construct macrostates to sample conformations and pathways, we do not need to pre-define an intractable number of macrostates. This is useful for relatively unknown or unfamiliar systems like the triazine trimer in Sec. III C.
2. Close to optimal macrostates are constructed, which have small statistical errors.
3. Computational cost is strictly controlled by reducing the aforementioned redundant walkers, while allowing the simulation to make progress in sampling the slowest process.

Furthermore, the CAS algorithm focuses on identifying critical pathways and transition states and is able to extract thermodynamic and kinetic information in a general setting. The CAS algorithm is also not hampered by the presence of metastable states since a constant stream of walkers at visited macrostates is maintained by resampling, an important feature from the WE method that has been improved upon by walkers having equal mean weights within each macrostate as stated in Sec. II A. Finally, as done in the WE method, we only need to tune a few parameters to increase efficiency in the sampling, such as simulation time, number of walkers per macrostate, and size of Voronoi cells.

ACKNOWLEDGMENTS

This work is supported by the Applied Mathematics Program within the Department of Energy (DOE) Office of Advanced Scientific Computing Research (ASCR) as part of the Collaboratory on Mathematics for Mesoscopic Modeling of Materials (CM4). We thank Hee Sun Lee for contributing to the regular WE method code that served as the basis for the CAS algorithm code, Johannes Birgmeier for making improvements on the parameter input file, Jeffery Kinnison for providing us with the penta-alanine simulation files, and Michael Daily for providing us with the triazine trimer simulation files. We also thank Jesús Izaguirre and the reviewers for giving us helpful comments that improved the presentation of the paper.

- ¹G. M. Torrie and J. P. Valleau, *J. Comput. Phys.* **23**, 187 (1977).
- ²A. Laio and M. Parrinello, *Proc. Natl. Acad. Sci. U. S. A.* **99**, 12562 (2002).
- ³A. Laio and F. L. Gervasio, *Rep. Prog. Phys.* **71**, 126601 (2008).
- ⁴A. Barducci, G. Bussi, and M. Parrinello, *Phys. Rev. Lett.* **100**, 020603 (2008).
- ⁵G. Bussi, A. Laio, and M. Parrinello, *Phys. Rev. Lett.* **96**, 090601 (2006).
- ⁶A. F. Voter, *Phys. Rev. Lett.* **78**, 3908 (1997).
- ⁷D. Hamelberg, J. Mongan, and J. A. McCammon, *J. Chem. Phys.* **120**, 11919 (2004).
- ⁸Y. Miao, F. Feixas, C. Eun, and J. A. McCammon, *J. Comput. Chem.* **36**, 1536 (2015).
- ⁹D. Rodríguez-Gómez, E. Darve, and A. Pohorille, *J. Chem. Phys.* **120**, 3563 (2004).
- ¹⁰D. Rodríguez-Gómez, E. Darve, and A. Pohorille, in *AbSciCon 2006* (Washington, DC, USA, 2006), pp. 26–30.
- ¹¹A. Pohorille and E. Darve, *AIP Conf. Proc.* **872**, 23–30 (2006).
- ¹²E. Darve and A. Pohorille, *J. Chem. Phys.* **115**, 9169 (2001).
- ¹³E. Darve, M. A. Wilson, and A. Pohorille, *Mol. Simul.* **28**, 113 (2002).
- ¹⁴E. Darve, D. Rodríguez-Gómez, and A. Pohorille, *J. Chem. Phys.* **128**, 144120 (2008).
- ¹⁵J. Comer, J. C. Gumbart, J. Hénin, T. Lelièvre, A. Pohorille, and C. Chipot, *J. Phys. Chem. B* **119**, 1129 (2015).
- ¹⁶Y. Sugita and Y. Okamoto, *Chem. Phys. Lett.* **314**, 141 (1999).
- ¹⁷M. R. So, A. F. Voter *et al.*, *J. Chem. Phys.* **112**, 9599 (2000).
- ¹⁸G. R. Bowman, V. S. Pande, and F. Noé, *An Introduction to Markov State Models and Their Application to Long Timescale Molecular Simulation* (Springer Science & Business Media, 2013), Vol. 797.
- ¹⁹G. R. Bowman, K. A. Beauchamp, G. Boxer, and V. S. Pande, *J. Chem. Phys.* **131**, 124101 (2009).
- ²⁰G. R. Bowman, X. Huang, and V. S. Pande, *Methods* **49**, 197 (2009).
- ²¹A. C. Pan and B. Roux, *J. Chem. Phys.* **129**, 064107 (2008).
- ²²C. Schütte, F. Noé, J. Lu, M. Sarich, and E. Vanden-Eijnden, *J. Chem. Phys.* **134**, 204105 (2011).
- ²³E. Suárez, J. L. Adelman, and D. M. Zuckerman, *J. Chem. Theory Comput.* **12**, 3473 (2016).
- ²⁴G. A. Huber and S. Kim, *Biophys. J.* **70**, 97 (1996).
- ²⁵B. W. Zhang, D. Jasnow, and D. M. Zuckerman, *J. Chem. Phys.* **132**, 054107 (2010).
- ²⁶D. Bhatt, B. W. Zhang, and D. M. Zuckerman, *J. Chem. Phys.* **133**, 014110 (2010).
- ²⁷B. W. Zhang, D. Jasnow, and D. M. Zuckerman, *Proc. Natl. Acad. Sci. U. S. A.* **104**, 18043 (2007).
- ²⁸E. Suarez, S. Lettieri, M. C. Zwier, C. A. Stringer, S. R. Subramanian, L. T. Chong, and D. M. Zuckerman, *J. Chem. Theory Comput.* **10**, 2658 (2014).
- ²⁹B. Abdul-Wahid, L. Yu, D. Rajan, H. Feng, E. Darve, D. Thain, and J. A. Izaguirre, in *2012 IEEE 8th International Conference on E-Science (IEEE, 2012)*, pp. 1–8.
- ³⁰R. Costaoeuc, H. Feng, J. Izaguirre, and E. Darve, *Discrete Contin. Dyn. Syst.* **2013**, 171.
- ³¹B. Abdul-Wahid, H. Feng, D. Rajan, R. Costaoeuc, E. Darve, D. Thain, and J. A. Izaguirre, *J. Chem. Inf. Model.* **54**, 3033 (2014).
- ³²J. A. Izaguirre, D. Thain, and E. Darve, in *SC15 Workshop: Producing High Performance and Sustainable Software for Molecular Simulation*, Austin, TX, 2015.
- ³³C. Trott, T.-R. Shan, S. Moore, A. Thompson, S. Plimpton, M. Höhnerbach, A. Ismail, P. Bientinesi, A. M. Elena, C. Lalanne *et al.*, in *Proceedings of the SC15 Workshop on Producing High Performance and Sustainable Software for Molecular Simulation*, 2016.
- ³⁴T. Schlick, *Innovations in Biomolecular Modeling and Simulations* (Royal Society of Chemistry, 2012), Vol. 1.
- ³⁵D. Aristoff, preprint [arXiv:1609.05887](https://arxiv.org/abs/1609.05887) (2016).
- ³⁶L. Maragliano, A. Fischer, E. Vanden-Eijnden, and G. Ciccotti, *J. Chem. Phys.* **125**, 024106 (2006).
- ³⁷E. Vanden-Eijnden, M. Venturoli, G. Ciccotti, and R. Elber, *J. Chem. Phys.* **129**, 174102 (2008).
- ³⁸M. C. Zwier, J. L. Adelman, J. W. Kaus, A. J. Pratt, K. F. Wong, N. B. Rego, E. Suárez, S. Lettieri, D. W. Wang, M. Grabe *et al.*, *J. Chem. Theory Comput.* **11**, 800 (2015).
- ³⁹D. Bhatt and I. Bahar, *J. Chem. Phys.* **137**, 104101 (2012).
- ⁴⁰J. L. Adelman and M. Grabe, *J. Chem. Phys.* **138**, 044105 (2013).
- ⁴¹A. Dickson and C. L. Brooks III, *J. Phys. Chem. B* **118**, 3532 (2014).
- ⁴²A. Dickson, A. M. Mustoe, L. Salmon, and C. L. Brooks, *Nucleic Acids Res.* **42**, 12126 (2014).
- ⁴³J.-H. Prinz, M. Held, J. C. Smith, and F. Noé, *Multiscale Model. Simul.* **9**, 545 (2011).
- ⁴⁴A. Y. Ng, M. I. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an algorithm,” in *Advances in Neural Information Processing (NIPS) 14*, edited by T. Dietterich, S. Becker, and Z. Ghahramani (MIT Press, Cambridge, MA, 2001).
- ⁴⁵U. Von Luxburg, *Stat. Comput.* **17**, 395 (2007).
- ⁴⁶R. T. McGibbon, B. E. Husic, and V. S. Pande, *J. Chem. Phys.* **146**, 044109 (2017).
- ⁴⁷J.-H. Prinz, J. D. Chodera, V. S. Pande, W. C. Swope, J. C. Smith, and F. Noé, *J. Chem. Phys.* **134**, 244108 (2011).
- ⁴⁸P. Metzner, F. Noé, and C. Schütte, *Phys. Rev. E* **80**, 021106 (2009).
- ⁴⁹H. Feng, R. Costaoeuc, E. Darve, and J. A. Izaguirre, *J. Chem. Phys.* **142**, 214113 (2015).
- ⁵⁰S. Pronk, S. Páll, R. Schulz, P. Larsson, P. Bjelkmar, R. Apostolov, M. R. Shirts, J. C. Smith, P. M. Kasson, D. van der Spoel *et al.*, *Bioinformatics* **29**, 845 (2013).
- ⁵¹J. Wang, R. M. Wolf, J. W. Caldwell, P. A. Kollman, and D. A. Case, *J. Comput. Chem.* **25**, 1157 (2004).
- ⁵²R. R. Coifman and S. Lafon, *Appl. Comput. Harmonic Anal.* **21**, 5 (2006).
- ⁵³J. de la Porte, B. M. Herbst, W. Hereman, and S. J. van der Walt, in *The 19th Symposium of the Pattern Recognition Association of South Africa* (Citeseer, 2008).
- ⁵⁴D. J. Tobias and C. L. Brooks III, *Biochemistry* **30**, 6059 (1991).
- ⁵⁵J. W. Grate, K.-F. Mo, and M. D. Daily, *Angew. Chem., Int. Ed.* **55**, 3925 (2016).
- ⁵⁶V. A. Voelz, K. A. Dill, and I. Chorny, *Pept. Sci.* **96**, 639 (2011).
- ⁵⁷E. A. Archer and M. J. Krische, *J. Am. Chem. Soc.* **124**, 5074 (2002).
- ⁵⁸M. Amm, N. Platzter, J. Guilhem, J. P. Bouchet, and J. P. Volland, *Magn. Reson. Chem.* **36**, 587 (1998).
- ⁵⁹H. E. Birkett, R. K. Harris, P. Hodgkinson, K. Carr, M. H. Charlton, J. C. Cherryman, A. M. Chippendale, and R. P. Glover, *Magn. Reson. Chem.* **38**, 504 (2000).