WESTPA 2.0: High-performance upgrades for weighted ensemble simulations and analysis of longer-timescale applications

John D. Russo[†,2], She Zhang[†,3], Jeremy M. G. Leung[†,1], Anthony T. Bogetti[†,1], Jeff P. Thompson[3], Alex J. DeGrave[1], Paul A. Torrillo[1], A. J. Pratt[1], Kim F. Wong[1], Junchao Xia[3], Jeremy Copperman[2], Joshua L. Adelman[5], Matthew C. Zwier[4], David N. LeBard[3], Daniel M. Zuckerman[2], and Lillian T. Chong[*,1]

[†] Equal authorship
* To whom correspondence should be addressed

[1]Department of Chemistry, University of Pittsburgh, Pittsburgh, PA; A.D. is presently at the Paul G. Allen School of Computer Science and Engineering, University of Washington, Seattle, WA; P.A.T. is presently at the Massachusetts Institute of Technology, Cambridge, MA
[2]Department of Biomedical Engineering, Oregon Health and Science University, Portland, OR
[3]OpenEye Scientific, Santa Fe, NM
[4]Department of Chemistry, Drake University, Des Moines, IA
[5]Department of Biological Sciences, University of Pittsburgh, Pittsburgh, PA

ABSTRACT

The weighted ensemble (WE) family of methods is one of several statistical-mechanics based path sampling strategies that can provide estimates of key observables (rate constants, pathways) using a fraction of the time required by direct simulation methods such as molecular dynamics or discrete-state stochastic algorithms. WE methods oversee numerous parallel trajectories using intermittent overhead operations at fixed time intervals, enabling facile interoperability with any dynamics engine. Here, we report on major upgrades to the WESTPA software package, an open-source, high-performance framework that implements both basic and recently developed WE methods. These upgrades offer substantial improvements over traditional WE. Key features of the new WESTPA 2.0 software enhance efficiency and ease of use: an adaptive binning scheme for more efficient surmounting of large free energy barriers, streamlined handling of large simulation datasets, exponentially improved analysis of kinetics, and developer-friendly tools for creating new WE methods, including a Python API and resampler module for implementing both binned and “binless” WE strategies.

1. Introduction

The field of molecular dynamics (MD) simulations of biomolecules arguably is following a trajectory that is typical of mathematical modeling efforts: as scientific knowledge grows, models grow ever more complex and ambitious, rendering them challenging for computation. While early MD simulations focused on single-domain small proteins,[1] modern simulations have attacked ever larger complexes[2,3] and even entire virus particles.[4–7] This trend belies the fact that record-setting small-protein simulations in terms of total simulation time remain limited to the ms scale on special-purpose resources[8] and to < 100 μs on typical university clusters. These limitations

have motivated the development of numerous approaches to accelerate sampling, among which are rigorous path-sampling approaches capable of providing unbiased kinetic and mechanistic observables.[9–18]

Our focus is the weighted ensemble (WE) path sampling approach,[17,19] which has helped to transform what is feasible for molecular simulations in the generation of pathways for long-timescale processes (> μs) with rigorous kinetics. Among these simulations are notable applications, including atomically detailed simulations of protein folding,[20] coupled protein folding and binding,[21] protein-protein binding,[22] protein-ligand unbinding,[23] and the large-scale opening of the SARS-CoV-2 spike protein.[24] The latter is a significant milestone—both in system size (half a million atoms) and timescale (seconds).[24] Instrumental to the success of the above applications have been advances in not only WE methods, but also software.[24]

Here, we present the next generation (version 2.0) of the most cited, open-source WE software called WESTPA (Weighted Ensemble Simulation Toolkit with Parallelization and Analysis).[25] WESTPA 2.0 is designed to further enhance the efficiency of WE simulations with high-performance algorithms for: (i) further enhanced sampling via restarting from reweighted trajectories, adaptive binning, and/or binless strategies, (ii) more efficient handling of large simulation datasets, and (iii) analysis tools for estimation of first-passage-time distributions and for more efficient estimation of rate constants. Like its predecessor, WESTPA 2.0 is a highly scalable, portable, and interoperable Python package that embodies the full range of WE's capabilities, including rigorous theory for any type of stochastic dynamics (e.g., molecular dynamics and Monte Carlo simulations) that is agnostic to the model resolution.[26] In comparison to other open-source WE packages such as AWE-WQ[27] and wepy,[28] WESTPA is unique in its (i) high scalability with nearly perfect scaling out to thousands of CPU cores[24] and GPUs, and (ii) demonstrated ability to interface with a variety of dynamics engines and model resolutions, including atomistic,[22] coarse-grained,[29] whole-cell,[30] and non-spatial systems models.[31,32]

After a brief overview of the WE strategy (**Section 2**), we describe the organization of WESTPA 2.0 (**Section 3**) and new analysis tools that further expand the capabilities of the software package (**Section 4**). Together, these features greatly facilitate the execution and analysis of WE simulations of even larger systems and/or slower timescales.

2. Overview of the weighted ensemble path sampling strategy

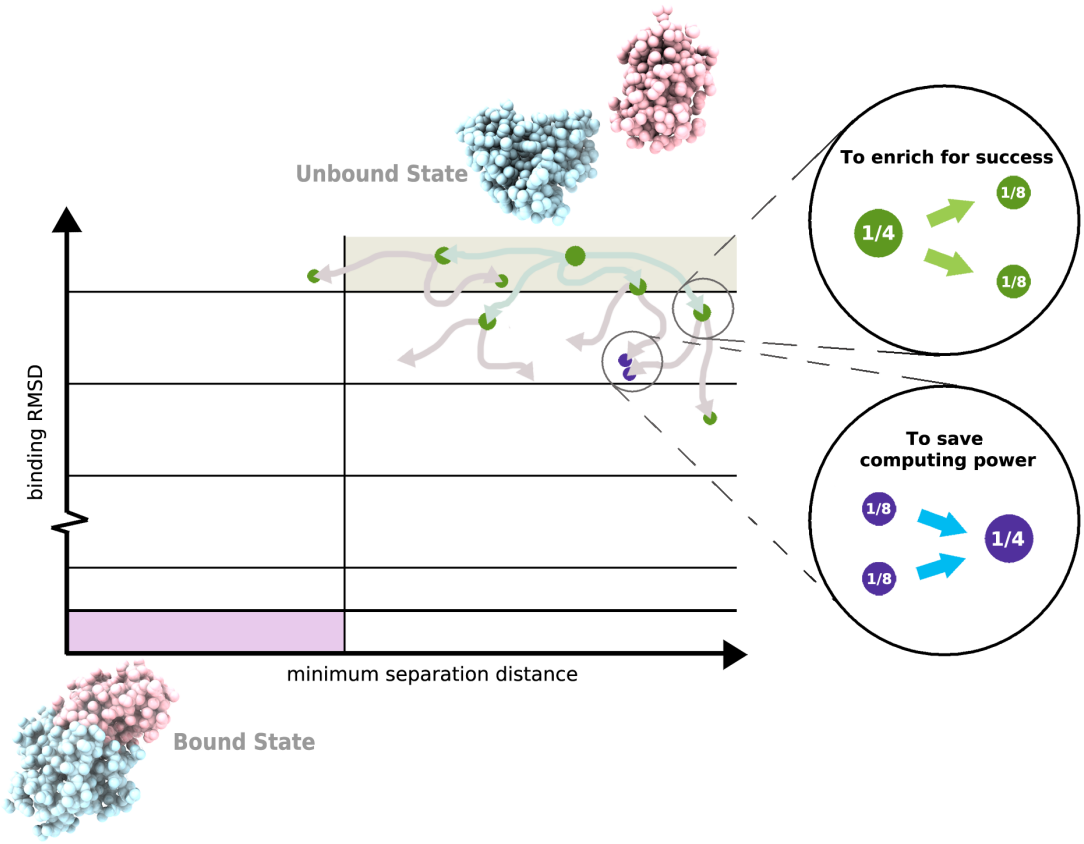

Figure 1. Basic weighted ensemble protocol. As illustrated for the simulation of a protein-protein binding process, a two-dimensional progress coordinate is divided into bins with the goal of occupying each bin with a target number of four trajectories. Four equally weighted trajectories are initiated from the unbound state and subjected to a resampling procedure at periodic time intervals τ: (i) to enrich for success, trajectories that make transitions to less-visited bins are replicated to generate a target of four trajectories in those bins, splitting the weights evenly among the child trajectories (green spheres), and (ii) to save computing time, the lowest-weight trajectories in bins that have exceeded four trajectories are terminated, merging their weights with those of higher-weight trajectories in those bins (purple spheres). Spheres are sized according to their statistical weights.

The weighted ensemble (WE) strategy enhances the sampling of rare events (e.g., protein folding, binding, chemical reactions) by orchestrating the periodic resampling of multiple, parallel trajectories at fixed time intervals τ (**Figure 1**).[17] The statistically rigorous resampling scheme maintains even coverage of configurational space by replicating ("splitting") trajectories that have made transitions to newly visited regions and potentially terminating ("merging") trajectories that have over-populated previously visited regions. The configurational space is typically defined by a progress coordinate that is divided into bins where even coverage of this space is defined as a constant number of trajectories occupying each bin; alternatively, trajectories may be grouped by a desired feature for "binless" resampling schemes.[33] Importantly, trajectories are assigned statistical weights that are rigorously tracked during resampling; when trajectories are replicated

in a given bin, the weights are split among child trajectories and when trajectories are terminated in a probabilistic fashion, the weights are merged with a continued trajectory of that bin. This rigorous tracking ensures that *no bias is introduced into the ensemble dynamics*, enabling direct estimates of rate constants.[26]

WE simulations can be run under equilibrium or non-equilibrium steady state conditions. To maintain non-equilibrium steady state conditions, trajectories that reach the target state are "recycled" back to the initial state, retaining the same statistical weight.[34] The advantage of equilibrium WE simulations over steady-state WE simulations is that the target state need not be strictly defined in advance since no recycling of trajectories at the target state is applied.[35] On the other hand, steady-state WE simulations have been more efficient in yielding successful pathways and estimates of rate constants. Equilibrium observables can be estimated from either equilibrium WE simulations or the combination of two non-equilibrium steady-state WE simulations in opposite directions when history information is taken into account.[35]

3. Organization of WESTPA 2.0

Below, we present the organization of WESTPA 2.0, beginning with code reorganization to facilitate software development (**Section 3.1**) and then proceeding to a description of a Python API for setting up, running, and analyzing WE simulations (**Section 3.2**); a minimal adaptive binning mapper (**Section 3.3**); a generalized resampler module that enables the implementation of both binned and binless schemes (**Section 3.4**); and an HDF5 framework for more efficient handling of large simulation datasets (**Section 3.5**).

3.1. Code reorganization to facilitate software development

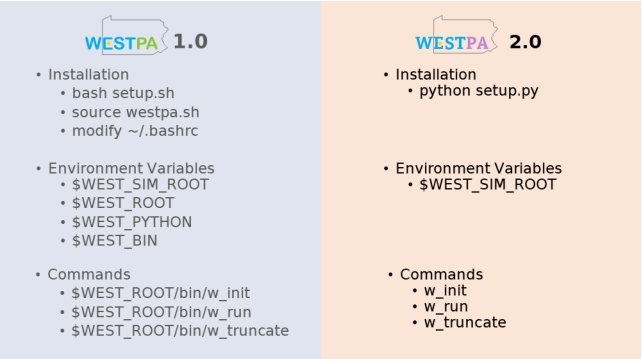

Figure 2. Reorganization of WESTPA 1.0 to WESTPA 2.0. In version 2.0, WESTPA is installed using Python and relies on only a single environment variable such that commands can be called directly through Python. To reflect these changes, we have updated our original suite of WESTPA tutorials for version 2.0 (https://github.com/westpa/westpa_tutorials/tree/westpa-2.0-restruct).[36,37]

The WESTPA 2.0 software is designed to facilitate the maintenance and further development of the software according to established and emerging best practices for Python development and

packaging. The code has been consolidated and reorganized to better indicate the role of each module (**Figure 2**). The software can now be installed as a standard Python package using pip or by running setup.py. The package will continue to be available through Conda via conda-forge, which streamlines the installation process by enabling WESTPA and all software dependencies to be installed at the same time. We have implemented automated GitHub Actions for continuous integration testing and code quality checks using the Black Python code formatter as a pre-commit hook, alongside flake8 for non-style linting. Templates are provided for GitHub issues and pull requests. Both user's and developer's guides are available on the GitHub wiki along with Sphinx documentation of key functions with autogenerated docstrings. Further support will continue to be provided through WESTPA users' and developers' email lists hosted on Google Groups (linked on https://westpa.github.io).

3.2 Python API for setting up, running, and analysis of WE simulations

To simplify the process of setting up and running WE simulations, WESTPA 2.0 features a Python API that enables the user to execute the relevant commands within a single Python script instead of invoking a series of command-line tools, as previously done in WESTPA 1.0 (**Figure 3A**). This also provides tools for third-party developers to build and develop WESTPA-based applications and plugins, for example, the integration of WESTPA into the cloud-based computing platform, OpenEye Scientific's Orion;[38,39] or the haMSM restarting plugin (**Section 4.2**), which uses the results of a WESTPA simulation to perform analysis then restart the simulation based on the results of that analysis.

Figure 3B provides an example of how to programmatically call the WESTPA 2.0 API from the Orion cloud platform, which could in principle be any Python script within any supercomputing or personal computing environment. First, a developer can write any custom simulation or work manager of their choice by subclassing or completely rewriting core WESTPA components (top panel). Second, a workflow can be constructed by invoking a simple set of WESTPA 2.0 Python commands to perform any WE simulation (bottom panel). Typically, a user of the WESTPA 2.0 Python API only needs a handful of API endpoints to perform a complicated simulation protocol. As an example of the power of the simplicity of the Python API, we demonstrate how a workflow can be constructed from defined workflow kernels (**Figure 3C**), and show GPU performance over wall-clock time (in Coordinated Universal Time; UTC) from a drug-like molecule in a membrane permeability simulation (**Figure 3D**). Using the internal API, a user's simulation can request large amounts of compute resources per iteration. In this case, thousands of GPUs are requested per WE iteration for a simulation of butanol crossing a natural membrane mimetic system (https://github.com/westpa/westpa2_tutorials).[40]

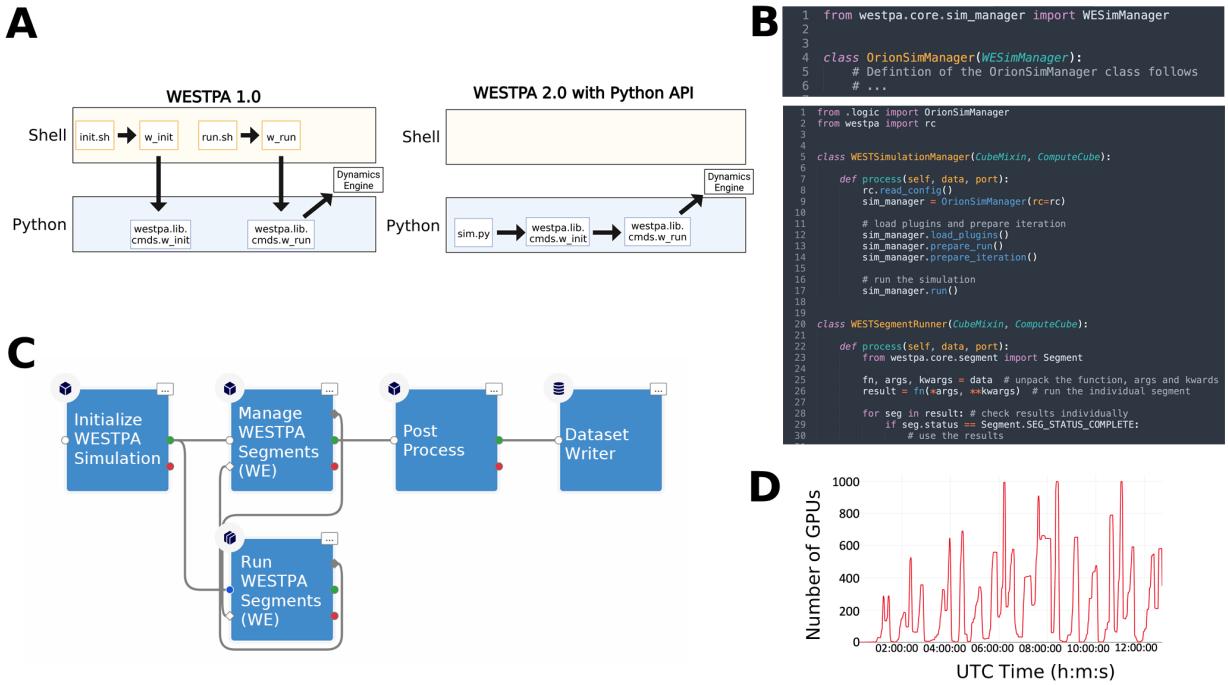

Figure 3. Comparison of workflows for setting up and running WE simulations using WESTPA 1.0 and 2.0, a demonstration of using the Python API for WESTPA 2.0, and GPU performance of the updated API within a cloud computing environment. (A) The Python API in WESTPA 2.0 enables a user to fully define, initialize, and run a WESTPA simulation from within a single Python script (right panel), without needing to invoke command-line utilities required in WESTPA 1.0 (left panel). For backwards compatibility, all original functionality provided in version 1.0 for invoking WESTPA (e.g., `w_init` and `w_run` tools) via shell scripts remains available in WESTPA 2.0. (B) Example of defining a custom simulation manager with the WESTPA 2.0 API (top panel), and using the newly defined simulation manager and WESTPA 2.0 API to programmatically control and run a WE simulation (bottom panel). Here, the WESTSimulationManager class sends work to the WESTSegmentRunner class that unpacks and runs the scripts specified from the WESTPA config file (west.cfg). (C) Example workflow diagram from the Orion user interface using the Python classes constructed from the internal WESTPA APIs presented in **Figure 3B**. Here, a kernel (Initialize WESTPA Simulation) initializes both the `WESTSimulationManager` (Manage WESTPA Segments) and the `WESTSimulationRunner` (Run WESTPA Segments) kernels from **Figure 3B**, which are connected in a cycle to manage splitting and merging. Finally, all data is exported through a Post Process and Dataset Writer kernel for final data processing and storage. (D) Performance of the WESTPA 2.0 API using the `WESTSimulationRunner` class from **Figure 3B** within an Amazon Web Services environment using a combination of numerous `g4dn` instances as a function of wallclock time in Universal Coordinated Time (UTC) units. Here, the per-iteration scaling reaches thousands of GPUs in just under a few hours for a test system of butanol crossing a neat POPC membrane bilayer using the WESTPA 2.0 API with the OpenMM 7.5 MD engine.[41]

To facilitate the development of custom analysis workflows in cases where more flexibility is required than the existing `w_ipa` analysis tool,[36] WESTPA 2.0 includes the new `westpa.analysis` Python API. This API provides a high-level view of the data contained in the main WESTPA HDF5 file ("west.h5"), including the trajectory data, reducing the overhead of writing custom analysis code in Python and doing quick, interactive analysis of trajectories (or walkers). The `westpa.analysis` API is built on three core data types: `Run`, `Iteration`, and `Walker`. A `Run` is a sequence of `Iterations`; an `Iteration` is a collection of `Walkers`. Key instance data can be accessed via attributes and methods. For example, a `Walker` has attributes such as the statistical weight (`weight`), progress coordinate value (`pcoords`), starting conformation (`parent`), and child trajectories after replication (`children`), and a method `trace()` to trace its history (as a pure Python alternative to the `w_trace` tool). The API also provides facilities for retrieving and concatenating trajectory segments. These include support for (i) type-aware concatenation of trajectory segments represented by NumPy arrays or MDTraj trajectories, (ii) use of multiple threads to potentially increase performance when segment retrieval is an I/O bound operation, and (iii) display of progress bars. Finally, the API provides a convenience function, `time_average()`, for computing the time average of an observable over a sequence of `Iterations` (e.g., all or part of a `Run`).

3.3. A minimal adaptive binning mapper

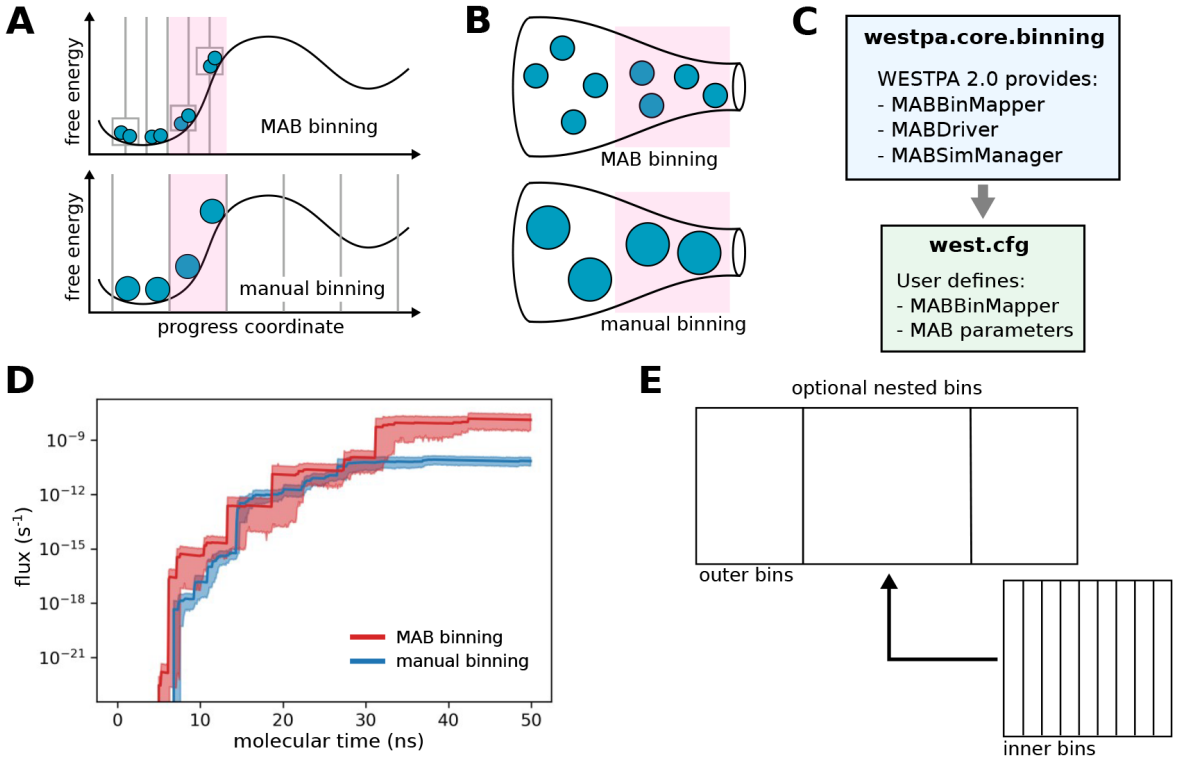

Figure 4. The minimal adaptive binning (MAB) scheme is more efficient in surmounting free energy barriers than manual, fixed binning schemes. (A) Bin positions and trajectories after

replication using the MAB scheme vs. a manual binning scheme with the same positions of trajectories (blue circles, sized according to statistical weights) along a chosen progress coordinate and a target of two trajectories per bin. The MAB scheme adaptively positions bins along the progress coordinate by placing equally spaced bins (in this case, three bins, as indicated by solid vertical lines) between the positions of the trailing and leading trajectories along with separate bins (boxes) for these trajectories and a third trajectory in a bottleneck region (pink) along the free energy barrier. (B) Enlarged "bottle" diagrams highlighting the bottleneck region (pink) along with relative positions and weights of trajectories for the MAB and manual binning schemes in panel A). In contrast to the manual binning scheme where trajectories may stall in a bottleneck region, the MAB scheme automatically detects trajectories in this region, replicating these trajectories to enrich for success in surmounting the barrier. (C) MAB-scheme options in the `westpa.core.binning` module and corresponding user-defined options in the west.cfg file. (D) Flux of a drug-like molecule (tacrine) permeating through a neat POPC membrane as a function of molecular time using fixed binning (blue) or adaptive binning (MAB scheme) (red). Solid lines represent mean fluxes and the shaded regions represent 95% confidence intervals. The molecular time is defined as Nτ, where N is the number of WE iterations and τ is the fixed time interval (100 ps) of each WE iteration. Simulations were run using WESTPA 2.0 and the OpenMM 7.5 MD engine.[41] (E) Schematic of a simple recursive binning case in which closely spaced inner bins are "nested" within a wider outer bin.

To automate the placement of bins along a chosen progress coordinate during WE simulation, we have implemented the Minimal Adaptive Binning (MAB) scheme[42] as an option in the `westpa.core.binning` module. The MAB scheme positions a specified number of bins along a progress coordinate after each resampling interval τ by (1) tagging the positions of the trailing and leading trajectories along the progress coordinate and evenly placing a specified number of bins between these positions, and (2) tagging "bottleneck" trajectories positioned on the steepest probability gradients and assigning these trajectories to their own bins (**Figures 4A-B**). Despite its simplicity, the MAB scheme requires less computing time than manual, fixed binning schemes in surmounting large free energy barriers resulting in more efficient conformational sampling and estimation of rate constants.[42] To apply the MAB scheme, users specify the `MABBinMapper` option along with accompanying parameters such as the number of bins in the west.cfg file (**Figure 4C**).

Figure 4D illustrates the effectiveness of the MAB scheme in enhancing the efficiency of simulating the membrane permeability of a drug-like molecule (tacrine). Relative to a fixed binning scheme, the MAB scheme results in earlier flux of tacrine through a model cellular membrane bilayer (~5 ns vs. ~7 ns) and this flux increases more quickly, achieving values that are two orders of magnitude higher for the duration of the test.

The MAB scheme provides a general framework for user creation of more complex adaptive binning schemes.[42] Users can now specify nested binning schemes in the west.cfg file (**Figure 4E**). To run WESTPA simulations under non-equilibrium steady-state conditions (i.e. with

"recycling" of trajectories that reach the target state) with the MAB scheme, users can nest a `MABBinMapper` inside of a `RecursiveBinMapper` bin and specify a target state as the outer bins. Multiple individual `MABBinMappers` can be created and placed at different locations of the outer bins using a recursive scheme, offering further flexibility in the creation of advanced binning schemes.

3.4. Generalized resampler module that enables binless schemes

In the original (default) weighted-ensemble resampling scheme, trajectories are split and merged based on a predefined set of bins.[17] In WESTPA 2.0, we introduce a generalized resampler module that enables users to implement both binned and "binless" resampling schemes, providing the flexibility to resample trajectories based on a property of interest by defining a grouping function. While grouping on the state last visited (e.g., initial or target state) was previously possible using the binning machinery in WESTPA 1.0,[43] our new resampler module provides a more general framework for creating binless schemes by defining a group/reward function of interest such as nonlinear progress coordinates that may be identified by machine learning techniques. Following others,[45] the resampler module includes options for (i) specifying a minimum threshold for trajectory weights to avoid running trajectories with inconsequentially low weights, and (ii) specifying a maximum threshold for trajectory weights to avoid a single large-weight trajectory from dominating the sampling, increasing the number of uncorrelated successful events that reach the target state.

As illustrated in **Figure 5**, the implementation of a binless scheme requires two modifications to the default WESTPA simulation: (i) a user-provided group module containing the methods needed to process the resampling property of interest for each trajectory walker, and (ii) updates to the west.cfg file specifying the resampling method in the `group_function` keyword and the attribute in the `group_arguments` keyword.

We provide two examples of implementing binless schemes in the westpa-2.0-restruct branch of the WESTPA_Tutorials GitHub repository (https://github.com/westpa/westpa_tutorials/tree/westpa-2.0-restruct).[37] The basic_nacl_group_by_history example illustrates grouping of trajectory based on its "history", i.e. a shared parent *N* WE iterations back. The parameter *N* is specified in the keyword hist_length under the `group_arguments` keyword in the west.cfg file. This WESTPA configuration file also specifies the name of the grouping function method, `group.walkers_by_history`, in the `group_function` keyword. In the basic_nacl_group_by_color example, trajectory walkers are tagged based on "color" according to the state last visited. Only walkers that have the same color are merged, thereby increasing the sampling of pathways in both directions. State definitions are declared within the `group_arguments` keyword in the west.cfg file.

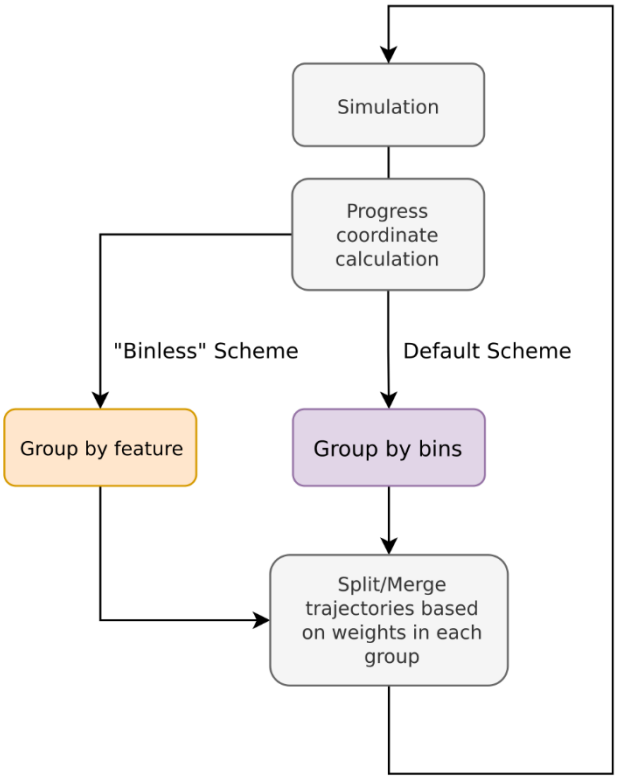

Figure 5. Flowchart for implementing binless resampling schemes in WESTPA 2.0. The implementation involves grouping trajectories by feature (using the `group_function` defined in the `group` module) before splitting and merging. The functionality for positioning bins along a chosen progress coordinate remains available in WESTPA 2.0.

3.5. HDF5 framework for more efficient handling of large simulation datasets

One major challenge of running WE simulations has been the management of the resulting large datasets, which can amount to tens of terabytes over millions of trajectory files. To address this challenge, we have developed a framework for storing trajectory data in a highly compressed and portable HDF5 file format. The format is derived from the HDFReporter class implemented in the MDTraj analysis suite,[46] and maintains compatibility with NGLView,[47] an iPython/Jupyter widget for interactive viewing of molecular structures and trajectories. A single HDF5 file is generated per WE iteration, which includes a link to each trajectory file stored in the main WESTPA data file (west.h5). Thus, the new HDF5 framework in WESTPA 2.0 enables users to restart a WE simulation from a single HDF5 file rather than millions of trajectory files and simplifies data sharing as well as analysis. The dramatic reduction in the number of trajectory files also eliminates potentially large overhead from the filesystem that results from the storage of numerous small files. For example, a 53% overhead has been observed for a 7.5-GB dataset of 103,260 trajectory files generated from NTL9 protein folding simulations (**Figure 9**), occupying 11.5 GB of actual disk storage on a Lustre filesystem.

To test the effectiveness of the HDF5 framework in reducing the amount of data storage required for WE simulations, we applied the framework to a set of three independent WE simulations of Na^+/Cl^- association and one WE simulation involving p53 peptide conformational sampling (**Figures 6A-B**). Our results revealed 27% and 85% average reduction in the total size of trajectory files generated during the Na^+/Cl^- association and p53 peptide simulations, respectively, relative to WESTPA 1.0. Given a fixed number of bins, the sizes of per-iteration HDF5 files were also shown to converge as the simulation progresses (**Figures 6C-D**), suggesting that the storage of trajectory data by iteration not only facilitates the management of the data but also yields files of roughly equal sizes. The difference in the reduction efficiency that we observed between the Na^+/Cl^- and p53 peptide systems can be attributed to differences in the simulation configurations including the format of the output trajectories, restart files and other factors such as the verbosity of logging.

Our tests revealed that the additional steps introduced by the HDF5 framework for managing trajectory coordinate and restart files did not have any significant impact on the WESTPA runtime (**Figure 6E**), which is normalized by the number of trajectory segments per WE iteration given that the evolution of bin occupancies by trajectories can vary among different runs due to the stochastic nature of the MD simulations (after 60 iterations, the WESTPA 1.0 run occupied six more bins than the WESTPA 2.0/HDF5 run). This variation in the bin occupancy is unlikely to be affected by the HDF5 framework since it simply manages the trajectory and restart files and does not alter how the system is simulated. The differences in bin occupancies/total number of trajectories may also partially contribute to the large reduction in per-iteration file sizes for the HDF5 run observed in **Figure 6D** of the p53 peptide. However, the majority of this file-size reduction results from efficient HDF5 data compression of trajectory coordinates, restart, and log files. Finally, trajectory data saved in the HDF5 files can be extracted and analyzed easily using MDTraj in combination with our new analysis framework presented in **Section 3.2** (**Figure 6F**).

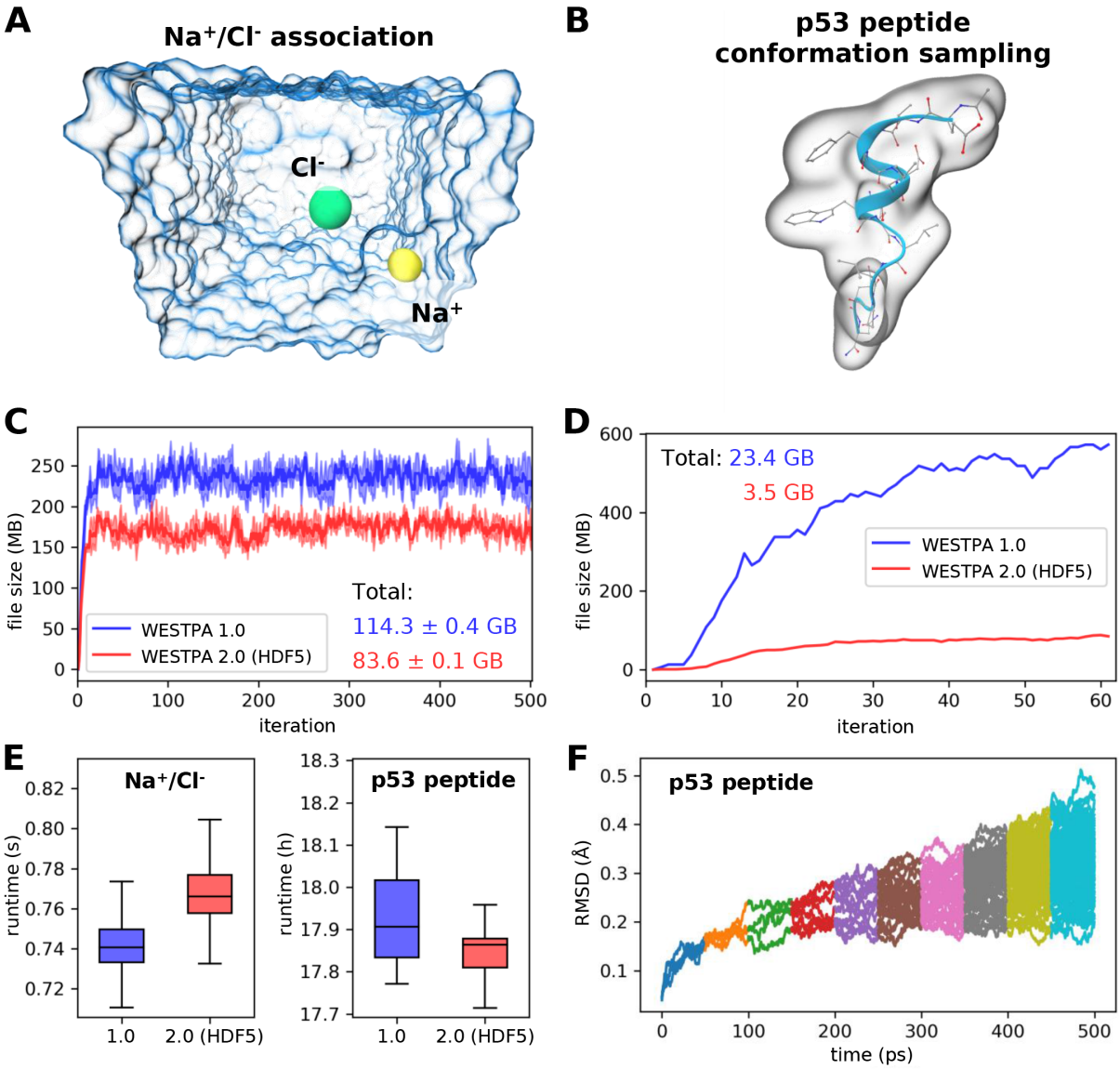

Figure 6. Demonstration of the usage of the HDF5 framework for two example systems. (A) Na+/Cl- association simulation where Na+ (*yellow sphere*) and Cl- (*green sphere*) ions were solvated in explicit water (*blue transparent surface*). The distance between the two ions serves as the progress coordinate. (B) Conformational sampling of a p53 peptide (residues 17-29) in generalized Born implicit solvent using a progress coordinate consisting of the heavy-atom RMSD of the peptide from its MDM2-bound conformation.[21] The molecular surface of the p53 peptide is rendered as a transparent surface, with both the secondary (*blue ribbon*) and atomic structures overlaid. (C) Comparison of file sizes of per-iteration HDF5 files for the Na^+/Cl^- association simulation as a function of the WE iteration using WESTPA 1.0 and 2.0 with the HDF5 framework. The result was obtained from three independent simulations where the *solid curves* show the mean file sizes, while the *light bands* show the standard deviations. (D) Same comparison as panel C for a single simulation of the p53 peptide, hence no error bars are shown. (E) Comparison of wall-clock runtimes normalized by the number of trajectory segments per WE

iteration using WESTPA 1.0 and 2.0 with the HDF5 framework option turned on. (F) Time-evolution of the heavy-atom RMSD of the p53 peptide from its MDM2-bound conformation by trajectories obtained using WESTPA's analysis tools. Colors represent RMSDs obtained from different iterations. WESTPA simulations of Na^+/Cl^- association and the p53 peptide were run using the OpenMM 7.5 MD engine.[41]

4. Analysis tools

WESTPA 2.0 features new analysis tools for estimating rate constants more efficiently using the distribution of "barrier crossing" times (**Section 4.1**), accelerating convergence using a history-augmented Markov state model to reweight trajectories (**Section 4.2**), and estimating the distribution of first passage times (**Section 4.3**).

4.1. The RED scheme for rate-constant estimation

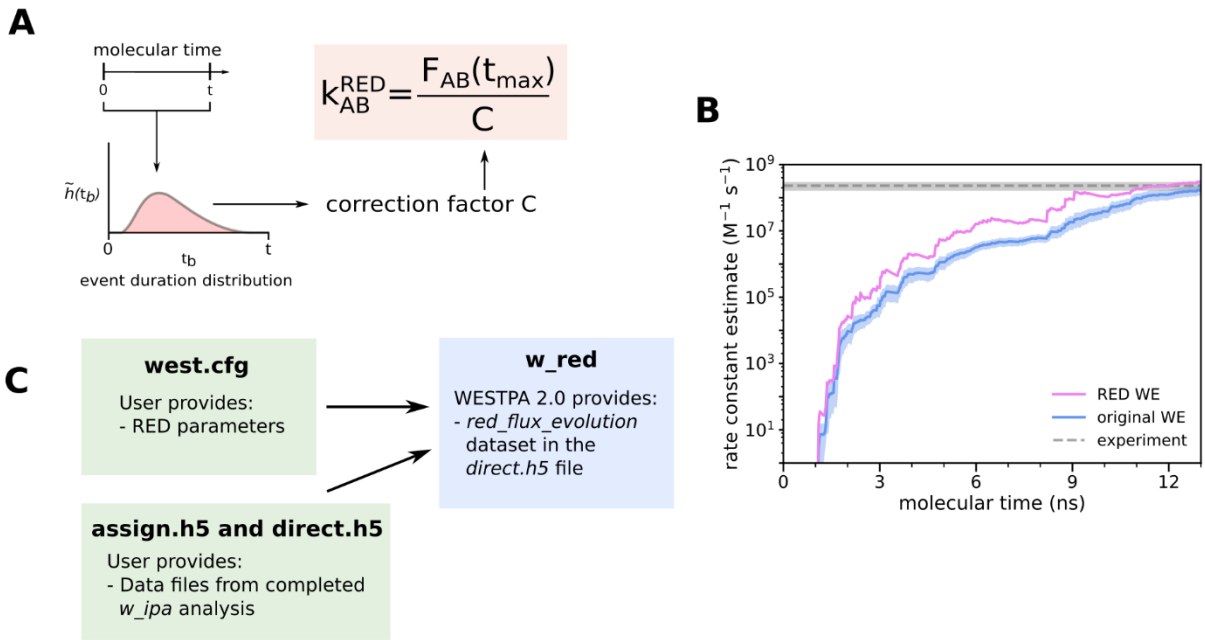

Figure 7. The RED scheme for more efficient rate-constant estimation. (A) Schematic illustrating the RED scheme, which incorporates the distribution of event durations as part of a correction factor for rate-constant estimates that accounts for statistical bias toward the observation of events with short durations. (B) Application of the original and RED schemes to estimate the associate rate constant of a protein-protein binding process involving the barnase and barstar proteins as a function of molecular time in a WE simulation. The molecular time is defined as Nτ, where N is the number of WE iterations and τ is the fixed time interval (20 ps) of each WE iteration. Simulations were previously run using WESTPA 1.0 with the GROMACS 4.6.7 MD engine.[48] (C) A schematic illustrating how users can generate a dataset for calculating the RED-scheme correction factor from simulation data stored in the analysis HDF5 files and apply the correction factor to the rate-constant estimate using the new `w_red` tool.

To more efficiently estimate rate constants from WE simulations, we have implemented the Rates from Event Durations (RED) scheme as an analysis tool called `w_red` in the WESTPA 2.0 software. The RED scheme exploits the transient ramp-up portion of a WE simulation by incorporating the probability distribution of event durations (or "barrier crossing" times) from a WE simulation as part of a correction factor (**Figure 7A**).[49] The correction factor accounts for the systematic error that results from statistical bias toward the observation of events with short durations and reweights the event duration distribution accordingly. When applied to an atomistic WE simulation of a protein-protein binding process, the RED scheme is >25% more efficient than the original WE scheme[17] in estimating the association rate constant (**Figure 7B**).[49]

The code for estimating rate constants using the RED scheme takes as input the assign.h5 files and direct.h5 files generated by the `w_ipa` analysis tool. Users then specify in the analysis section of the west.cfg file which analysis scheme `w_red` should analyze along with the initial/final states and the number of frames per iteration. Executing `w_red` from the command line, will output the corrected flux estimates as a new dataset called red_flux_evolution to the users' existing direct.h5 file (**Figure 7C**). The RED rate-constant estimates can then be accessed through the Python h5py module and plotted vs. time to assess the convergence of the estimates. To estimate uncertainties in observables calculated from a small number of trials (i.e. number of independent WE simulations), we recommend using the Bayesian Bootstrap approach.[17,50] If it is not feasible to run multiple independent simulations with a certain system due to either system size or the timescale of the process of interest, a user can apply a Monte Carlo bootstrapping approach to a single simulation's RED rate constant estimate.

4.2. A history-augmented Markov State Model (haMSM) restarting plugin

History-augmented Markov state models (haMSMs) provide a powerful tool for estimation of stationary distributions and rate constants from transient, unconverged WE data.[51] Thus, the approach has a similar motivation to the RED scheme.[49] In haMSM analysis instead of discretizing trajectories via the WE bins used by WESTPA, as in the `WESS/WEED` reweighting plugins,[34,35] a much finer and more numerous set of 'microbins' is employed to calculate steady-state properties with higher accuracy. These estimates, in turn, can be used to start new WE simulations from a steady-state estimate, accelerating convergence of the simulation.[50] The new plugin provides a streamlined implementation of the restarting protocol that runs automatically as part of a WESTPA simulation, a capability which did not previously exist.

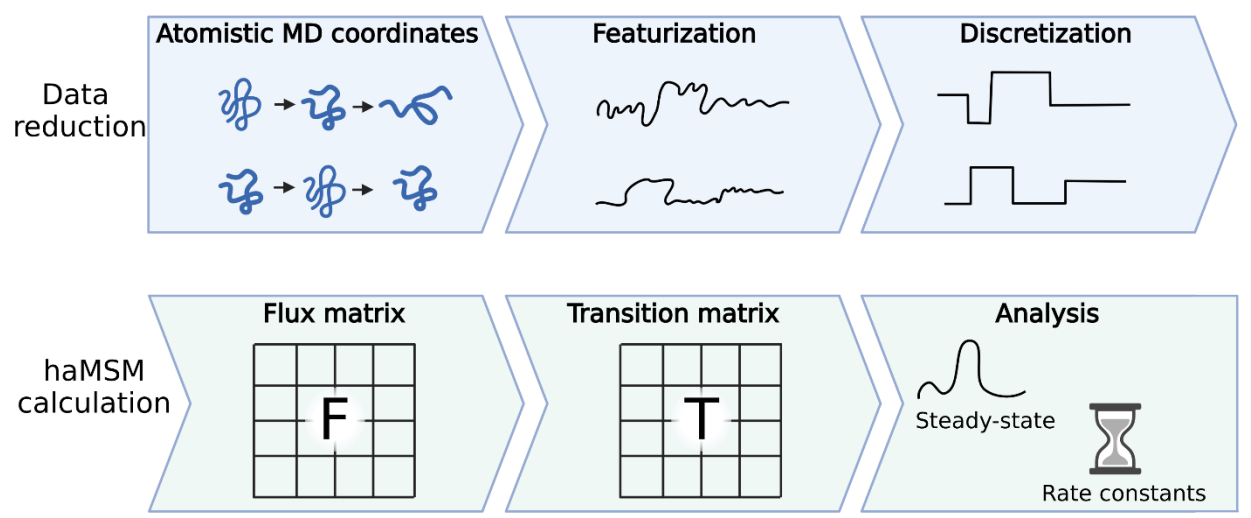

Figure 8. Workflow for constructing an haMSM from trajectories. First, the atomistic trajectories are featurized and discretized. The flux matrix is then computed by computing fluxes between discrete states. The flux matrix is row-normalized into a transition matrix. Estimates of steady-state populations and rate constants are obtained from analysis of the transition matrix.[52]

The `msm_we` package provides a set of analysis tools for using typical WESTPA HDF5 output files, augmented with atomic coordinates, to construct an haMSM. A nearly typical MSM model-building procedure[53] is used (**Figure 8**): WE trajectories are discretized into clusters (microbins) and transitions among microbins are analyzed. However, instead of reconstructing entire trajectories, the `msm_we` analysis computes the flux matrix by taking each weighted parent/child segment pair, extracting and discretizing one frame from each, and measuring flux between them - i.e. the weight transferred.

The haMSM restarting plugin in WESTPA 2.0 makes use of the analysis tools provided by `msm_we` to incorporate this functionality directly into WESTPA. It manages running a number of independent simulations, initialized from some starting configuration, and augments their output HDF5 with the necessary atomic coordinates. Data from all independent runs are gathered and used to build a single haMSM. Stationary probability distributions and rate constants are estimated from this haMSM.

This plugin can be used to start a set of new WE simulation runs, initialized closer to steady-state (**Figure 9**). The haMSM and the WE trajectory data are used to build a library of structures and their associated steady-state weights. These are used to initiate a new set of independent WE runs, which should start closer to steady-state and thus converge more quickly. The process can be repeated iteratively, as shown in **Figure 9A**. The result of this restarting procedure is shown in **Figure 9B**. For challenging systems, the quality of the haMSM will greatly affect the quality of the steady-state estimate. A further report is forthcoming on strategies for building high-quality haMSMs.

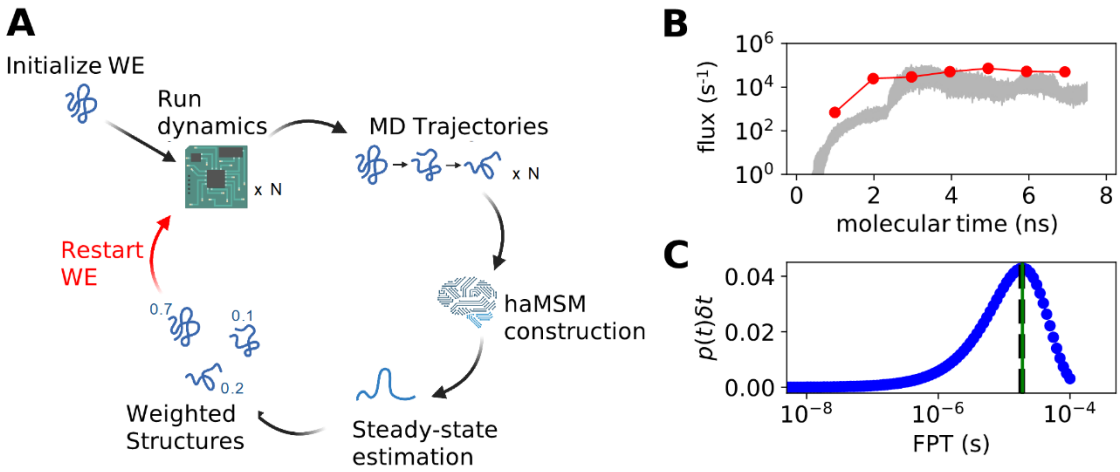

Figure 9. Application of haMSM restarting plugin to the ms folding process of the NTL9 protein. (A) Diagram of the haMSM restarting plugin's functionality. (B) Example of restarting plugin functionality in accelerated convergence of NTL9 folding rate constants from a WESTPA 2.0 simulation using the AMBER 16 MD engine.[54] haMSM estimates at restarting points are shown as dots, WE direct fluxes are shown as red lines, and a 95% credibility region from direct WE is shown in gray. (C) Distribution of first passage times for NTL9 folding from the haMSM built at the final restart of the simulation in **Figure 9B.** The weighted average of the blue FPT distribution is shown in black dashed, and the MFPT estimate from the haMSM's steady-state estimate is shown in green.[52]

To use this plugin, users must specify a function that ingests coordinate data and featurizes the data. Dimensionality reduction may be performed on this featurized data. An effective choice of featurization provides a more granular structural description of the system without including a large number of irrelevant coordinates that add noise without adding useful information. For example, a limited subset of the full atoms such as only alpha-carbons, or even a strided selection of the alpha carbons, may be sufficient to capture the important structural information. Choosing a featurization based on rotation-invariant distances, such as pairwise atomic distances instead of atomic positions, can also help capture structural fluctuations without sensitivity to large-scale motion of the entire system.

To validate convergence of the restarted simulations, a number of independent replicates of the restarting protocol should be performed. These replicates should demonstrate both stability in flux estimates across restarts, and relatively constant-in-time direct fluxes within the restarts. If limited to a single replicate, agreement between the haMSM flux estimate and the direct flux should also be validated.

4.3. Estimating first-passage-time distributions

First passage times (FPTs) and their mean values (MFPTs) are key kinetics quantities to characterize many stochastic processes (from a macrostate to another) in chemistry and biophysics such as chemical reactions, ligand binding and unbinding, protein folding, diffusion processes of small molecules within crowded environments. WE simulations, via the Hill relation, provide unbiased estimates of the MFPT directly once steady is reached[34] or indirectly via non-Markovian haMSM analysis,[35] but mathematically rigorous estimation of the FPT distribution is not available and has been a challenge for WE simulation. Suárez and coworkers, however, have shown that the FPT distributions estimated from haMSM models provide semi-quantitative agreement with unbiased reference distributions in different systems.[55] Details on building history-augmented MSMs are described above in **Section 4.2** and more information can be found in the references.[35,55]

Here, we extend and strengthen earlier FPT distribution analysis from WE data. The original code for calculating FPT distribution was published on a separate GitHub repository (https://github.com/ZuckermanLab/NMpathAnalysis).[56] Recently we reorganized and refactored the code in class hierarchical structures: a base class (`MatrixFPT`) for calculating MFPTs and FPTs distribution using a general transition matrix as an input parameter, and two derived classes (`MarkovFPT` and `NonMarkovFPT`) using transition matrices from Markovian analysis and non-Markovian analysis such as haMSM in **Section 4.2** respectively. The updated code has been merged into the `msm_we` package described in **Section 4.2** along with some updates on building transition matrix from classic MD simulation trajectories.

The new code enables robust estimation of the FPT distribution. **Figure 9C** shows the non-Markovian estimation of the FPT distribution of transitions between macrostate A and B from the WE simulation of NTL9 protein folding.

5. Summary

WESTPA is an open-source, high-scalable, interoperable software package for applying the weighted ensemble (WE) strategy, which greatly increases the efficiency of simulating rare events (e.g., protein folding, protein binding) while maintaining rigorous kinetics. The latest WESTPA release (version 2.0) is a substantial upgrade from the original software with high-performance algorithms enabling the simulation of ever more complex systems and processes and implementing new analysis tools. WESTPA 2.0 has also been reorganized into a more standard Python package to facilitate code development of new WE algorithms, including binless strategies. With these features available in the WESTPA toolbox, the WE community is well-poised to take advantage of the latest strategies for tackling major challenges in rare-events sampling, including the identification of slow coordinates using machine learning techniques,[57,58] and the interfacing of the WE strategy with other software involving complementary rare-event sampling strategies (e.g., OpenPathSampling,[59,60] SAFFIRE,[61] and ScMile[62]) and analysis tools (e.g., LOOS,[63,64] MDAnalysis,[65,66] and PyEmma[67]). WESTPA has also been interfaced with OpenEye Scientific's Orion platform[39] on the Amazon Web Services cloud computing facility. We

hope that the above new features of WESTPA will greatly facilitate efforts by the scientific community to tackle grand challenges in the simulation of rare events in a variety of fields, including the molecular sciences and systems biology.

Notes

The authors declare the following competing financial interest(s). L.T.C. is a current member of the Scientific Advisory Board of OpenEye Scientific and an Open Science Fellow with Roivant Sciences. S.Z., J.P.T., J.X., and D.N.L. are employees of OpenEye Scientific.

Acknowledgments

This work was supported by an NIH grant (R01 GM115805) to L.T.C. and D.M.Z.; NSF grants (CHE-1807301 and MCB-2112871) to L.T.C.; a MolSSI Software Fellowship to J.D.R.; and a University of Pittsburgh Andrew Mellon Graduate Fellowship to A.T.B. Computational resources were provided by the University of Pittsburgh's Center for Research Computing, by OpenEye Scientific via compute instances sourced from Amazon Web Services, and by the Advanced Computing Center at Oregon Health and Science University. We thank David Aristoff, Gideon Simpson, Forrest York, Darian Yang, and Alan Grossfield for helpful discussions.

References

(1) McCammon, J. A.; Gelin, B. R.; Karplus, M. Dynamics of Folded Proteins. *Nature* **1977**, *267* (5612), 585–590.

(2) Casalino, L.; Gaieb, Z.; Goldsmith, J. A.; Hjorth, C. K.; Dommer, A. C.; Harbison, A. M.; Fogarty, C. A.; Barros, E. P.; Taylor, B. C.; McLellan, J. S.; Fadda, E.; Amaro, R. E. Beyond Shielding: The Roles of Glycans in the SARS-CoV-2 Spike Protein. *ACS Cent. Sci.* **2020**, *6* (10), 1722–1734.

(3) Anandakrishnan, R.; Zhang, Z.; Donovan-Maiye, R.; Zuckerman, D. M. Biophysical Comparison of ATP Synthesis Mechanisms Shows a Kinetic Advantage for the Rotary Process. *PNAS* **2016**, *113* (40), 11220–11225.

(4) Zhao, G.; Perilla, J. R.; Yufenyuy, E. L.; Meng, X.; Chen, B.; Ning, J.; Ahn, J.; Gronenborn, A. M.; Schulten, K.; Aiken, C.; Zhang, P. Mature HIV-1 Capsid Structure by Cryo-Electron Microscopy and All-Atom Molecular Dynamics. *Nature* **2013**, *497* (7451), 643–646.

(5) Perilla, J. R.; Schulten, K. Physical Properties of the HIV-1 Capsid from All-Atom Molecular Dynamics Simulations. *Nat Commun* **2017**, *8* (1), 15959.

(6) Casalino, L.; Dommer, A. C.; Gaieb, Z.; Barros, E. P.; Sztain, T.; Ahn, S.-H.; Trifan, A.; Brace, A.; Bogetti, A. T.; Clyde, A.; Ma, H.; Lee, H.; Turilli, M.; Khalid, S.; Chong, L. T.; Simmerling, C.; Hardy, D. J.; Maia, J. D.; Phillips, J. C.; Kurth, T.; Stern, A. C.; Huang, L.; McCalpin, J. D.; Tatineni, M.; Gibbs, T.; Stone, J. E.; Jha, S.; Ramanathan, A.; Amaro, R. E. AI-Driven Multiscale Simulations Illuminate Mechanisms of SARS-CoV-2 Spike Dynamics. *The International Journal of High Performance Computing Applications* **2021**, *35* (5), 432–451.

(7) Jung, J.; Nishima, W.; Daniels, M.; Bascom, G.; Kobayashi, C.; Adedoyin, A.; Wall, M.; Lappala, A.; Phillips, D.; Fischer, W.; Tung, C.-S.; Schlick, T.; Sugita, Y.; Sanbonmatsu, K. Y. Scaling Molecular Dynamics beyond 100,000 Processor Cores for Large-Scale Biophysical Simulations. *Journal of Computational Chemistry* **2019**, *40* (21), 1919–1930.

(8) Shaw, D. E.; Maragakis, P.; Lindorff-Larsen, K.; Piana, S.; Dror, R. O.; Eastwood, M. P.; Bank, J. A.; Jumper, J. M.; Salmon, J. K.; Shan, Y.; Wriggers, W. Atomic-Level Characterization of the Structural Dynamics of Proteins. *Science* **2010**, *330* (6002), 341–346.
(9) Zuckerman, D. M.; Woolf, T. B. Efficient Dynamic Importance Sampling of Rare Events in One Dimension. *Phys. Rev. E* **2000**, *63* (1), 016702.
(10) Faradjian, A. K.; Elber, R. Computing Time Scales from Reaction Coordinates by Milestoning. *J. Chem. Phys.* **2004**, *120* (23), 10880–10889.
(11) West, A. M. A.; Elber, R.; Shalloway, D. Extending Molecular Dynamics Time Scales with Milestoning: Example of Complex Kinetics in a Solvated Peptide. *J Chem Phys* **2007**, *126* (14), 145104.
(12) van Erp, T. S.; Moroni, D.; Bolhuis, P. G. A Novel Path Sampling Method for the Calculation of Rate Constants. *J. Chem. Phys.* **2003**, *118* (17), 7762–7774.
(13) Allen, R. J.; Valeriani, C.; Wolde, P. R. ten. Forward Flux Sampling for Rare Event Simulations. *J. Phys.: Condens. Matter* **2009**, *21* (46), 463102.
(14) Pratt, L. R. A Statistical Method for Identifying Transition States in High Dimensional Problems. *J. Chem. Phys.* **1986**, *85* (9), 5045–5048.
(15) Swenson, D. W. H.; Bolhuis, P. G. A Replica Exchange Transition Interface Sampling Method with Multiple Interface Sets for Investigating Networks of Rare Events. *J. Chem. Phys.* **2014**, *141* (4), 044101.
(16) DeFever, R. S.; Sarupria, S. Contour Forward Flux Sampling: Sampling Rare Events along Multiple Collective Variables. *J. Chem. Phys.* **2019**, *150* (2), 024103.
(17) Huber, G. A.; Kim, S. Weighted-Ensemble Brownian Dynamics Simulations for Protein Association Reactions. *Biophysical Journal* **1996**, *70* (1), 97–110.
(18) Ray, D.; Stone, S. E.; Andricioaei, I. *Markovian Weighted Ensemble Milestoning (M-WEM): Long-Time Kinetics from Short Trajectories*; 2021; p 2021.06.26.450057.
(19) Zuckerman, D. M.; Chong, L. T. Weighted Ensemble Simulation: Review of Methodology, Applications, and Software. *Annu Rev Biophys* **2017**, *46*, 43–57.
(20) Adhikari, U.; Mostofian, B.; Copperman, J.; Subramanian, S. R.; Petersen, A. A.; Zuckerman, D. M. Computational Estimation of Microsecond to Second Atomistic Folding Times. *J. Am. Chem. Soc.* **2019**, *141* (16), 6519–6526.
(21) Zwier, M. C.; Pratt, A. J.; Adelman, J. L.; Kaus, J. W.; Zuckerman, D. M.; Chong, L. T. Efficient Atomistic Simulation of Pathways and Calculation of Rate Constants for a Protein–Peptide Binding Process: Application to the MDM2 Protein and an Intrinsically Disordered P53 Peptide. *J. Phys. Chem. Lett.* **2016**, *7* (17), 3440–3445.
(22) Saglam, A. S.; Chong, L. T. Protein–Protein Binding Pathways and Calculations of Rate Constants Using Fully-Continuous, Explicit-Solvent Simulations. *Chem. Sci.* **2019**, *10* (8),
(23) Lotz, S. D.; Dickson, A. Unbiased Molecular Dynamics of 11 Min Timescale Drug Unbinding Reveals Transition State Stabilizing Interactions. *J. Am. Chem. Soc.* **2018**, *140* (2), 618–628.
(24) Sztain, T.; Ahn, S.-H.; Bogetti, A. T.; Casalino, L.; Goldsmith, J. A.; Seitz, E.; McCool, R. S.; Kearns, F. L.; Acosta-Reyes, F.; Maji, S.; Mashayekhi, G.; McCammon, J. A.; Ourmazd, A.; Frank, J.; McLellan, J. S.; Chong, L. T.; Amaro, R. E. A Glycan Gate Controls Opening of the SARS-CoV-2 Spike Protein. *Nat. Chem.* **2021**, 1–6.
(25) Zwier, M. C.; Adelman, J. L.; Kaus, J. W.; Pratt, A. J.; Wong, K. F.; Rego, N. B.; Suárez, E.; Lettieri, S.; Wang, D. W.; Grabe, M.; Zuckerman, D. M.; Chong, L. T. WESTPA: An Interoperable, Highly Scalable Software Package for Weighted Ensemble Simulation and Analysis. *J. Chem. Theory Comput.* **2015**, *11* (2), 800–809.

(26) Zhang, B. W.; Jasnow, D.; Zuckerman, D. M. The "Weighted Ensemble" Path Sampling Method Is Statistically Exact for a Broad Class of Stochastic Processes and Binning Procedures. *J. Chem. Phys.* **2010**, *132* (5), 054107.
(27) Abdul-Wahid, B.; Feng, H.; Rajan, D.; Costaouec, R.; Darve, E.; Thain, D.; Izaguirre, J. A. AWE-WQ: Fast-Forwarding Molecular Dynamics Using the Accelerated Weighted Ensemble. *J. Chem. Inf. Model.* **2014**, *54* (10), 3033–3043.
(28) Lotz, S. D.; Dickson, A. Wepy: A Flexible Software Framework for Simulating Rare Events with Weighted Ensemble Resampling. *ACS Omega* **2020**, *5* (49), 31608–31623.
(29) Saglam, A. S.; Chong, L. T. Highly Efficient Computation of the Basal K_on Using Direct Simulation of Protein–Protein Association with Flexible Molecular Models. *J. Phys. Chem. B* **2016**, *120* (1), 117–122.
(30) Donovan, R. M.; Tapia, J.-J.; Sullivan, D. P.; Faeder, J. R.; Murphy, R. F.; Dittrich, M.; Zuckerman, D. M. Unbiased Rare Event Sampling in Spatial Stochastic Systems Biology Models Using a Weighted Ensemble of Trajectories. *PLOS Computational Biology* **2016**, *12* (2), e1004611.
(31) Tapia, J.-J.; Saglam, A. S.; Czech, J.; Kuczewski, R.; Bartol, T. M.; Sejnowski, T. J.; Faeder, J. R. MCell-R: A Particle-Resolution Network-Free Spatial Modeling Framework. *Methods Mol Biol* **2019**, *1945*, 203–229.
(32) Johnson, M. E.; Chen, A.; Faeder, J. R.; Henning, P.; Moraru, I. I.; Meier-Schellersheim, M.; Murphy, R. F.; Prüstel, T.; Theriot, J. A.; Uhrmacher, A. M. Quantifying the Roles of Space and Stochasticity in Computer Simulations for Cell Biology and Cellular Biochemistry. *MBoC* **2021**, *32* (2), 186–210.
(33) Donyapour, N.; Roussey, N. M.; Dickson, A. REVO: Resampling of Ensembles by Variation Optimization. *J Chem Phys* **2019**, *150* (24), 244112.
(34) Bhatt, D.; Zhang, B. W.; Zuckerman, D. M. Steady-State Simulations Using Weighted Ensemble Path Sampling. *The Journal of Chemical Physics* **2010**, *133* (1), 014110.
(35) Suárez, E.; Lettieri, S.; Zwier, M. C.; Stringer, C. A.; Subramanian, S. R.; Chong, L. T.; Zuckerman, D. M. Simultaneous Computation of Dynamical and Equilibrium Information Using a Weighted Ensemble of Trajectories. *J. Chem. Theory Comput.* **2014**, *10* (7), 2658–2667.
(36) Bogetti, A. T.; Mostofian, B.; Dickson, A.; Pratt, A. J.; Saglam, A. S.; Harrison, P. O.; Dudek, M.; Torrillo, P. A.; DeGrave, A. J.; Adhikari, U.; Zweir, M. C.; Zuckerman, D. M.; Chong, L. T. A Suite of Tutorials for the WESTPA Rare-Events Sampling Software [Article v1.0]. *Living Journal of Computational Molecular Science* **2019**, *1* (2), 10607–10607.
(37) WESTPA. *WESTPA Tutorials*; Github Repository, 2021. https://github.com/westpa/westpa_tutorials.
(38) *Orion*; OpenEye Scientific Software: Santa Fe, NM.
(39) Grebner, C.; Malmerberg, E.; Shewmaker, A.; Batista, J.; Nicholls, A.; Sadowski, J. Virtual Screening in the Cloud: How Big Is Big Enough? *J. Chem. Inf. Model.* **2020**, *60* (9), 4274–4282.
(40) WESTPA. *WESTPA 2.0 Tutorials*; Github Repository, 2021. https://github.com/westpa/westpa2_tutorials.
(41) Eastman, P.; Swails, J.; Chodera, J. D.; McGibbon, R. T.; Zhao, Y.; Beauchamp, K. A.; Wang, L.-P.; Simmonett, A. C.; Harrigan, M. P.; Stern, C. D.; Wiewiora, R. P.; Brooks, B. R.; Pande, V. S. OpenMM 7: Rapid Development of High Performance Algorithms for Molecular Dynamics. *PLOS Computational Biology* **2017**, *13* (7), e1005659.
(42) Torrillo, P. A.; Bogetti, A. T.; Chong, L. T. A Minimal, Adaptive Binning Scheme for Weighted Ensemble Simulations. *J. Phys. Chem. A* **2021**, *125* (7), 1642–1649.

(43) Adelman, J. L.; Grabe, M. Simulating Rare Events Using a Weighted Ensemble-Based String Method. *J. Chem. Phys.* **2013**, *138* (4), 044105.
(45) Dickson, A.; Brooks, C. L. WExplore: Hierarchical Exploration of High-Dimensional Spaces Using the Weighted Ensemble Algorithm. *J. Phys. Chem. B* **2014**, *118* (13), 3532–3542.
(46) McGibbon, R. T.; Beauchamp, K. A.; Harrigan, M. P.; Klein, C.; Swails, J. M.; Hernández, C. X.; Schwantes, C. R.; Wang, L.-P.; Lane, T. J.; Pande, V. S. MDTraj: A Modern Open Library for the Analysis of Molecular Dynamics Trajectories. *Biophysical Journal* **2015**, *109* (8), 1528–1532.
(47) Nguyen, H.; Case, D. A.; Rose, A. S. NGLview–Interactive Molecular Graphics for Jupyter Notebooks. *Bioinformatics* **2018**, *34* (7), 1241–1242.
(48) Hess, B.; Kutzner, C.; van der Spoel, D.; Lindahl, E. GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation. *J. Chem. Theory Comput.* **2008**, *4* (3), 435–447.
(49) DeGrave, A. J.; Bogetti, A. T.; Chong, L. T. The RED Scheme: Rate-Constant Estimation from Pre-Steady State Weighted Ensemble Simulations. *J. Chem. Phys.* **2021**, *154* (11), 114111.
(50) *Numerical Recipes in C: The Art of Scientific Computing*; Press, W. H., Flannery, B. P., Teukolsky, S. A., Vetterling, W. T., Eds.; Cambridge University Press: Cambridge, 1988.
(51) Copperman, J.; Zuckerman, D. M. Accelerated Estimation of Long-Timescale Kinetics from Weighted Ensemble Simulation via Non-Markovian "Microbin" Analysis. *J. Chem. Theory Comput.* **2020**, *16* (11), 6763–6775.
(52) *Created with Biorender.Com*; 2021.
(53) Chodera, J. D.; Noé, F. Markov State Models of Biomolecular Conformational Dynamics. *Curr Opin Struct Biol* **2014**, *25*, 135–144.
(54) Salomon-Ferrer, R.; Case, D. A.; Walker, R. C. An Overview of the Amber Biomolecular Simulation Package. *WIREs Computational Molecular Science* **2013**, *3* (2), 198–210.
(55) Suárez, E.; Pratt, A. J.; Chong, L. T.; Zuckerman, D. M. Estimating First-Passage Time Distributions from Weighted Ensemble Simulations and Non-Markovian Analyses. *Protein Science* **2016**, *25* (1), 67–78.
(56) ZuckermanLab. *NMpathAnalysis*; Github Repository, 2021. https://github.com/ZuckermanLab/NMpathAnalysis.
(57) Bhowmik, D.; Gao, S.; Young, M. T.; Ramanathan, A. Deep Clustering of Protein Folding Simulations. *BMC Bioinformatics* **2018**, *19* (18), 484.
(58) Ribeiro, J. M. L.; Bravo, P.; Wang, Y.; Tiwary, P. Reweighted Autoencoded Variational Bayes for Enhanced Sampling (RAVE). *J. Chem. Phys.* **2018**, *149* (7), 072301.
(59) Swenson, D. W. H.; Prinz, J.-H.; Noe, F.; Chodera, J. D.; Bolhuis, P. G. OpenPathSampling: A Python Framework for Path Sampling Simulations. 1. Basics. *J. Chem. Theory Comput.* **2019**, *15* (2), 813–836.
(60) Swenson, D. W. H.; Prinz, J.-H.; Noe, F.; Chodera, J. D.; Bolhuis, P. G. OpenPathSampling: A Python Framework for Path Sampling Simulations. 2. Building and Customizing Path Ensembles and Sample Schemes. *J. Chem. Theory Comput.* **2019**, *15* (2), 837–856.
(61) DeFever, R. S.; Hanger, W.; Sarupria, S.; Kilgannon, J.; Apon, A. W.; Ngo, L. B. Building A Scalable Forward Flux Sampling Framework Using Big Data and HPC. In *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning)*; PEARC '19; Association for Computing Machinery: New York, NY, USA, 2019; pp 1–8.

(62) Wei, W.; Elber, R. ScMile: A Script to Investigate Kinetics with Short Time Molecular Dynamics Trajectories and the Milestoning Theory. *J Chem Theory Comput* **2020**, *16* (2), 860–874.
(63) Romo, T. D.; Grossfield, A. LOOS: An Extensible Platform for the Structural Analysis of Simulations. In *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*; 2009; pp 2332–2335.
(64) Romo, T. D.; Leioatts, N.; Grossfield, A. Lightweight Object Oriented Structure Analysis: Tools for Building Tools to Analyze Molecular Dynamics Simulations. *Journal of Computational Chemistry* **2014**, *35* (32), 2305–2318.
(65) Michaud-Agrawal, N.; Denning, E. J.; Woolf, T. B.; Beckstein, O. MDAnalysis: A Toolkit for the Analysis of Molecular Dynamics Simulations. *Journal of Computational Chemistry* **2011**, *32* (10), 2319–2327.
(66) Gowers, R. J.; Linke, M.; Barnoud, J.; Reddy, T. J. E.; Melo, M. N.; Seyler, S. L.; Domański, J.; Dotson, D. L.; Buchoux, S.; Kenney, I. M.; Beckstein, O. MDAnalysis: A Python Package for the Rapid Analysis of Molecular Dynamics Simulations. *Proceedings of the 15th Python in Science Conference* **2016**, 98–105.
(67) Scherer, M. K.; Trendelkamp-Schroer, B.; Paul, F.; Pérez-Hernández, G.; Hoffmann, M.; Plattner, N.; Wehmeyer, C.; Prinz, J.-H.; Noé, F. PyEMMA 2: A Software Package for Estimation, Validation, and Analysis of Markov Models. *J. Chem. Theory Comput.* **2015**, *11* (11), 5525–5542.

Table of Contents/Abstract Image

For the manuscript "WESTPA 2.0: High-performance upgrades for weighted ensemble simulations and analysis of longer-timescale applications" by Russo *et al.*