

Temporal Subspace Clustering for Molecular Dynamics Data

Anna Beer*
 University of Vienna
 Vienna, Austria
 anna.beer@univie.ac.at

Martin Heinrigs*
 LMU Munich
 Munich, Germany
 heinrigs-martin@t-online.de

Claudia Plant
 University of Vienna
 Vienna, Austria
 plant@univie.ac.at

Ira Assent
 Aarhus University
 Aarhus, Denmark
 ira@cs.au.dk

ABSTRACT

We introduce MOSCITO (MOlecular Dynamics Subspace Clustering with Temporal Observance), a subspace clustering for molecular dynamics data. MOSCITO groups those timesteps of a molecular dynamics trajectory together into clusters in which the molecule has similar conformations. In contrast to state-of-the-art methods, MOSCITO takes advantage of sequential relationships found in time series data. Unlike existing work, MOSCITO does not need a two-step procedure with tedious post-processing, but directly models essential properties of the data. Interpreting clusters as Markov states allows us to evaluate the clustering performance based on the resulting Markov state models. In experiments on 60 trajectories and 4 different proteins, we show that the performance of MOSCITO achieves state-of-the-art performance in a novel single-step method. Moreover, by modeling temporal aspects, MOSCITO obtains better segmentation of trajectories, especially for small numbers of clusters.

KEYWORDS

Clustering, Molecular Dynamics, Subspace Clustering

1 INTRODUCTION

Proteins play an important role in every living organism. Analyzing the transition between different shapes of a protein, also known as folding, is important for understanding the structure and function of proteins, e.g., for drug design. Many diseases like Alzheimer, Parkinson, and different types of cancer are connected to misfolding of certain proteins [7]. Thus, understanding such folding processes may help to prevent or cure diseases.

Molecular dynamics data contains trajectories of proteins that change their shape over time. It is usually very high-dimensional, with thousands of time steps and hundreds of atoms for a single protein. Analyzing molecular dynamics data with traditional clustering is typically not meaningful due to the curse of dimensionality. To mitigate issues with high dimensionality, subspace clustering methods detect clusters in lower-dimensional subspaces [8]. However, existing subspace clustering methods do not cater for the characteristics of molecular dynamics data.

In this paper, we introduce MOSCITO (**M**OLEcular **D**ynamics **S**ubspace **C**lustering with **T**emporal **O**bservance), a subspace clustering algorithm that uses temporal regularization to exploit sequential relationships found in molecular dynamics trajectories that describe folding activities of single molecules. MOSCITO is the first one-step approach to handle this complex data type by identifying the best subspace projection. It finds features tailored specifically to molecular dynamics data. Leveraging methods from video analytics,

we show that MOSCITO finds temporally contiguous groups of protein conformations.

In trajectories, data points that are near in time are usually more closely related than those that are further apart. MOSCITO allows users to choose different sizes of time windows and different weighting methods for temporal regularization: binary, Gaussian, exponential, or logarithmic. Interestingly, the resulting clusters can be interpreted as states in a Markov State Model (MSM), which we use for comparing MOSCITO to approaches like SSC, PCA + k-Means, and TICA + k-Means. As we demonstrate in thorough empirical evaluation, MOSCITO successfully clusters molecular dynamics data in a model that directly captures the essential features and temporal relationships. Our main contributions are as follows:

- We introduce MOSCITO, a subspace clustering method for molecular dynamics data that makes use of temporal relations between time steps that are close
- MOSCITO finds temporally contiguous groups of protein conformations that are similar
- In contrast to currently used methods that require several steps, the clusters found by MOSCITO can *directly* be used as states of a Markov State Model describing the trajectory, yielding state-of-the-art quality with a one-in-all holistic approach

2 CLUSTERING TRAJECTORIES OF MOLECULAR DYNAMICS DATA

In this paper, we study the trajectories of a single molecule over time. A trajectory is given by the three-dimensional Cartesian coordinates of all atoms of the molecule for a number of time steps with fixed intervals. Where most traditional clustering methods target numerical data in \mathbb{R}^d , clustering MD data brings some unique challenges as we outline in the following.

Due to the size of MD data, automated methods for analyzing them are required. These often involve clustering methods to automatically group similar groups of atoms or timesteps.

In this paper, we focus on simulations of very long molecular dynamics trajectories that show folding processes of single molecules [25]. Based on the starting positions of a molecule's atoms, physical forces on all other atoms can be calculated, yielding simulated trajectories with a temporal resolution that usually lies in the femtosecond range. While the overall shape of the molecules is decisive for its function, several features are shown to be relevant, e.g., the torsions between C^α atoms that form the backbone of the protein, distances between specific atoms, or the solvent accessible surface area.

There are different use cases for which a clustering of MD trajectories is required. E.g., clusters of atoms that move similarly over time can be used to find dynamic domains, i.e., fragments of a protein that are internally rigid [19]. Clustering time steps in

*Both authors contributed equally to this research.

which the conformation of the protein, i.e., the arrangement of its atoms in space, is similar is used for detecting relevant states of the protein. Those states can be linked to (local) valleys in the energy landscape of the protein and are e.g. used for developing Markov models that describe a protein trajectory. Thus, one goal is to separate metastable states of a protein and the transitions between those states.

3 MARKOV STATE MODELING

Markov state models (MSMs) can be used to describe the dynamics of a MD trajectory by using a $n \times n$ transition matrix. To generate an MSM from an MD trajectory $X = \{x_1, \dots, x_t\}$, discrete states of the trajectory need to be identified. State-of-the-art methods use a two-step approach in which they first reduce the dimensionality of the data by using PCA or TICA and then clustering it with k-Means into n discrete states $S = \{S_1, \dots, S_n\}$ [35]. These states are used to calculate the transition probability matrix for a given lag time τ by counting the transitions from time step t to $t + \tau$:

$$p_{ij}(\tau) = \mathbb{P}(x_{t+\tau} \in S_j | x_t \in S_i) \quad (1)$$

The resulting MSM can be used for further analysis of the trajectory. In this paper, we analyze the clustering performance of our method in comparison to state-of-the-art methods for clustering MD trajectories based on their suitability as Markov states.

The variational approach for Markov models (VAMP) [37] is a generalization of the *Variational approach to conformational dynamics* (VAC) [14], [13] and provides a tool to evaluate MSMs. The VAC allows the approximation of eigenvalues and eigenfunctions with statistical observables [13]. It also defines a family of scoring functions, called the *VAMP – r*, which can be calculated from data. This score can be used to compare the choices of hyper-parameters, like the featurization of the data or the used clustering, used for constructing the MSM. The score is calculated from the singular value decomposition of the Koopman operator. The Koopman operator is explained below. It is important to note that the VAMP score can only be compared for MSMs constructed with the same lag time τ and cannot be used to find the optimal lag time τ itself.

The Koopman operator gives the conditional expected value of an arbitrary observable g at time $t + \tau$ for a given x_t . The Koopman operator K_τ of a Markov process is defined as follows [37]:

$$K_\tau g(x) = \mathbb{E}[g(x_{t+\tau} | x_t = x)] \quad (2)$$

When choosing the Dirac delta function δ_y centered at y , applying the Koopman operator evaluates the transition density of the dynamics [37].

$$K(\tau) = C_{00}^{-\frac{1}{2}} C_{01} C_{11}^{-\frac{1}{2}} \approx U_m \Sigma_m V_m^T, \quad (3)$$

where $\Sigma_m = \text{diag}(\sigma_1, \dots, \sigma_m)$ are the first m singular values.

VAMP-r Score. The VAMP-score [37] defines a function that can be used to score the quality of MSMs for a set lag-time τ . This score can be used to compare the methods used to create the discrete trajectories from the same featured trajectory [27].

With Σ_m as defined above (see Equation (3)), the VAMP-r score can be written as:

$$\text{VAMP-}r = \sum_{i=1}^m \sigma_i^r \quad (4)$$

The VAMP-2-score is based on the sum of squared eigenvalues of the transition matrix and can be interpreted as kinetic content [27]. In this paper, we use the VAMP-2 score to score the performance of different discretization methods for MD trajectories since there usually is no given ground truth to compare to. The most commonly used scores are the VAMP-1, which is analogous to the Rayleigh trace, and the VAMP-2, which is analogous to the kinetic variance [21].

4 SUBSPACE CLUSTERING

In high-dimensional datasets where the curse of dimensionality renders traditional clustering infeasible, subspace clustering finds clusters in a union of lower dimensional subspaces [18]. For a dataset $X = [x_1, x_2, \dots, x_n] \in \mathbb{R}^d$ it is often assumed that vectors x_j are drawn from a union of k subspaces $\{S_i\}_{i=1}^k$ of unknown dimensionality [9]. As MD data is typically very high dimensional, subspace clustering is a suitable approach. For details on subspace clustering, see [16, 32]. In this paper, we focus on spectral-based subspace clustering.

The advantages of spectral subspace clustering methods include a relatively simple implementation and robustness regarding data corruption and initialization [18]. Spectral subspace clustering methods usually perform two steps. First, a subspace representation of the original data is learned and used to construct an affinity matrix. Then spectral clustering is used on the affinity matrix to obtain the final clustering. Many different spectral algorithms like sparse subspace clustering (SSC) [4] or Low-Rank Representation (LRR) [10] have been proposed, for an overview see [32]. While spectral clustering has been successfully applied to numeric data, it lacks the ability to handle temporal data. We, thus, turn our attention to methods that have been developed with a different application goal in mind, but, as we argue, with similarities that make them attractive for the design of our MD subspace clustering method.

Temporal subspace clustering (TSC) [9] is a subspace clustering algorithm designed to segment video sequences into separate human motions. Video sequences are high-dimensional time series, with one frame of the video at each timestep. Significant changes in a video sequence usually are not immediate, but take place over many frames. Neighboring time frames are more correlated to each other than more distant frames. In that sense, molecular dynamics data is similar to video data. Unlike classical subspace clustering methods, which do not take any relations between the dimensions into account, TSC is designed to take advantage of the sequential relationship in time series data. TSC uses a dictionary learning approach together with a temporal Laplacian regularization function to encode those relationships in the dataset. Dictionary learning is a method that learns a set of basis vectors, a so-called dictionary [30]. The original data can then be represented as linear combinations of the basis vectors. TSC combines the dictionary learning approach with a temporal regularization function to encode the sequential relationships in the data. A non-negative dictionary and a corresponding coding are learned from the given data creating a robust affinity graph.

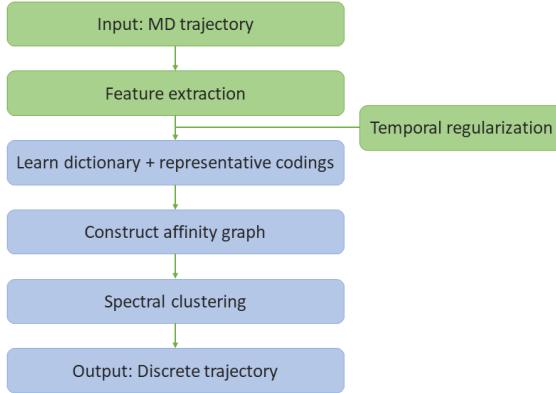


Figure 1: MOSCITO derives MD features from input trajectories. Temporal regularization provides the means for effective subspace clustering with temporal relations.

5 MOSCITO: TEMPORAL SUBSPACE CLUSTERING FOR MD DATA

We are now ready to describe our MOSCITO (MOlecular dynamics Subspace Clustering with Temporal Observance) approach, the first subspace clustering method for molecular dynamics data that captures temporal relations. Existing methods [35] often rely on a two- or three-step approach consisting of 1) dimensionality reduction, 2) clustering, and 3) finding macrostates by combining several clusters. As these steps are handled in isolation, their separate results may not be optimal overall. To avoid such undesirable side effects, MOSCITO takes a holistic approach: Firstly, in subspace clustering, it inherently combines clustering with dimensionality reduction. Secondly, MOSCITO incorporates temporal relation by weighting sequential neighbors in a time window. In contrast, existing subspace clustering algorithms cannot capture e.g., correlation between consecutive time steps.

For our new method MOSCITO, we exploit the similarity between MD data and video data (see Section 4) as well as the advances in the field of data mining on videos: TSC [9] clusters video data while incorporating the temporal relations between time steps as well as allowing clusters in subspaces of the data. Thus, we base MOSCITO on TSC. However, TSC is not able to handle MD data, because despite some similarities between the videos and MD data (e.g., smooth temporal transitions between time steps and potential clusters in subspaces), they also have strong differences: for every time step t , MD data describes a chain of atoms with its 3d coordinates, while video data contains colors for every pixel, where pixels are in a 2d grid. Thus, we need to change several steps and include knowledge from the field of MD as shown in Figure 1, which provides an overview of MOSCITO. The blue parts are equivalent to those of TSC [9], see Section 7. The green parts are adapted or changed completely. At first, essential MD features are extracted from the MD trajectory. From the features and a temporal regularization function, we learn a dictionary and the corresponding representation coding matrix. We then construct an affinity graph, and generate temporal segments by clustering the affinity graph.

Feature extraction. From the input MD trajectors, we extract features that capture essential properties.

Cartesian coordinates: The C^α atoms determine the shape and function of a protein, thus we use the common approach to focus solely on their 3d coordinates, rather than considering all atoms. For each frame, coordinates are centered and aligned to the first frame of the trajectory.

Backbone torsions: The backbone torsions determine the main structure of a protein. The torsion angles ϕ and ψ describe the dihedral angles between atoms of the backbone. Backbone torsions better describe the structure of a protein than absolute coordinates [31] as internal angles are independent of the global rotation of the protein, removing the need to align all frames to a fixed rotation.

Distance based: Heavy atoms are all atoms that are not hydrogen. We extract the minimal distance between heavy atoms [2] (we exclude the pairs at indices $(i, i+1)$ and $(i, i+2)$, as their distance is largely determined by their chemical bonds). Following [21], distances d are transformed to the negative exponential function e^{-d} .

Flexible torsions: The flexible torsion feature consists of the χ_1 to χ_5 side chain torsions. The side chains of a protein are chains of atoms, connected to the backbone. The angles are transformed into sine and cosine angles.

Solvent accessible surface area: For each frame of the trajectory, the solvent accessible surface area (SASA) of each residue is calculated using the Shrake-Rupley algorithm [26]. The SASA of a residue is calculated by summing up the SASA of each of the residues atoms

3D shape histogram: Based on [1], a combined 3D shape histogram for each frame of the trajectory is created. The state space is separated into five equally spaced shells around the origin of the coordinate system. The shells are split up further by combining them with 12 sectors into 60 cells. Histograms are created by counting the number of atoms in each cell. Unlike [1], all atoms instead of only the surface atoms were used as we are not only interested in the shape and function of the protein at a certain time step, but also in its internal folded structure. Final histograms are normalized between 0 and 1.

Temporal regularization. For trajectories, data points at neighboring time steps are usually more closely related than those at more distant time steps. Let the i -th column of the coding matrix Z a representation of vector x_i , then the encodings of its sequential neighbors, e.g., z_{i-1}, z_{i+1} should be similar to z_i . To achieve this, we use a temporal Laplacian regularization function $f(Z)$.

$$f(Z) = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} \|z_i - z_j\|_2^2 = \text{tr}(Z L_T Z^T) \quad (5)$$

L_T is the temporal Laplacian matrix defined as $L_T = \tilde{D} - W$. W is the weight matrix, capturing the sequential relationship in X , and $\tilde{D}_{ii} = \sum_{j=1}^n w_{ij}$ is a diagonal matrix with the sum of the rows of W on its diagonal.

The temporal regularization in MOSCITO weighs the sequential neighbors using a weight matrix $W \in \mathbb{R}^{n \times n}$ for a trajectory with n timesteps. Row i of W represents the weights of x_i 's neighbors. In the base case, binary weights are used. This weighs all s sequential neighbors of x_i the same, independent of the actual temporal

distance between the data points. In contrast to TSC, MOSCITO offers several naturally, points in close temporal proximity are more strongly correlated than points further apart in time. To model the decrease in correlation, we introduce different decaying weightings:

- Binary:** weights of 1 for points in the neighborhood and 0 outside
- Gaussian:** Gaussian curve fitted to the neighborhood
- Logarithmic:** Logarithmic decay with a value of 1 for the direct neighbors dropping logarithmically to 0 for x_{i+s} and x_{i-s}
- Exponential:** Exponential decay w.r.t. the temporal distance

Dictionary learning. Similar to TSC [9], MOSCITO learns to represent dataset X by a dictionary D and a coding matrix Z , such that $X \approx DZ$ [9].

Adopting a least square regression approach to dictionary learning, we use the following objective function [9]

$$\begin{aligned} & \min_{Z,D} \|X - DZ\|_F^2 + \lambda_1 \|Z\|_F^2 \\ & \text{s.t. } Z \geq 0, D \geq 0, \|d_i\|_2^2 \leq 1, \end{aligned} \quad (6)$$

where $\|Z\|_F^2$ denotes the Frobenius norm, and we use λ_1 for balancing both terms.

Importantly, MOSCITO integrates temporal regularization to capture relationships between nearby time steps. We achieve this by adding temporal regularization function $f(Z)$ such that our new objective function is the same as in TSC [9]

$$\begin{aligned} & \min_{Z,D} \|X - DZ\|_F^2 + \lambda_1 \|Z\|_F^2 + \lambda_2 f(Z) \\ & \text{s.t. } Z \geq 0, D \geq 0, \|d_i\|_2^2 \leq 1, \end{aligned} \quad (7)$$

with λ_2 tradeoff parameter [9].

To solve the dictionary learning problem efficiently, we follow the alternating direction method of multipliers (ADMM) [9]. It decomposes the problem into subproblems that are more easy to solve. Using auxiliary variables U, V , Eq. 7 is equivalent to

$$\begin{aligned} & \min_{Z,D,U,V} \|X - UV\|_F^2 + \lambda_1 \|V\|_F^2 + \lambda_2 f(V) \\ & \text{s.t. } U = D, V = Z, Z \geq 0, D \geq 0, \|d_i\|_2^2 \leq 1. \end{aligned} \quad (8)$$

The augmented Lagrangian of Eq. 8 is:

$$\begin{aligned} \mathcal{L} = & \frac{1}{2} \|X - UV\|_F^2 + \lambda_1 \|V\|_F^2 + \lambda_2 \text{tr}(V L_T V^T) \\ & + \langle \Lambda, U \rangle + \langle \Pi, V - Z \rangle + \frac{\alpha}{2} \|U - D\|_F^2 + \frac{\beta}{2} \|V - Z\|_F^2, \end{aligned} \quad (9)$$

with Λ and Π Lagrangian multipliers [9], which can now be solved by alternately minimizing \mathcal{L} respective to V, U, Z and D .

Updating V: To find the minimum of \mathcal{L} , its derivative with respect to V is set to zero, resulting in the following equation [9]:

$$[I \otimes (U^T U + (\lambda_1 + \alpha) I) + \lambda_2 L_T \otimes I] \text{vec}(V) = \text{vec}(U^T X - \Pi + \alpha Z) \quad (10)$$

Updating U: As for V , set derivative of \mathcal{L} wrt. U to zero [9]:

$$U = (X V^T - \Lambda + \alpha D) (V V^T + \alpha I)^{-1} \quad (11)$$

Updating Z and D: The update rules for Z and D are [9]:

$$Z = F_+ \left(V + \frac{\Pi}{\beta} \right), \quad D = F_+ \left(U + \frac{\Lambda}{\alpha} \right) \quad (12)$$

where $(F_+(A))_{ij} = \max\{A_{ij}, 0\}$, non-negativity constraint. Additionally, each column of D is normalized to unit length $\|d_i\|_2^2 \leq 1$.

Clustering. Using the learned coding matrix Z , an affinity graph is constructed. To capture sequential relationships, the following similarity measure is used [9]:

$$G(i, j) = \frac{z_i^T z_j}{\|z_i\|_2 \|z_j\|_2} \quad (13)$$

Spectral clustering on graph G yields the final clusters.

Algorithm 1: MOSCITO

```

input : Trajectory traj, dictionary size d, sequential
        neighbors s, number of clusters k, parameters  $\lambda_1$ ,
         $\lambda_2, \alpha, \beta$ 
output: Discrete trajectory vector d_traj

1 Extract features  $X$  from traj;
2 while not converged do
3   Update  $V_{k+1}$  according to Eq. 10 while fixing others;
4   Update  $U_{k+1}$  according to Eq. 11 while fixing others;
5   Update  $Z_{k+1}$  according to Eq. 12 while fixing others;
6   Update  $D_{k+1}$  according to Eq. 12 while fixing others;
7   Update  $\Pi_{k+1} : \Pi_{k+1} = \Pi_k + v\alpha(V_{k+1} - Z_{k-1})$ ;
8   Update  $\Lambda_{k+1} : \Lambda_{k+1} = \lambda_k + v\beta(U_{k+1} - D_{k-1})$ ;
9    $k = k + 1$ ;
10 end while
11 Generate affinity graph  $G$  using Eq. 13;
12 Perform spectral clustering on  $G$ , get d_traj;
```

6 EXPERIMENTS

Implementation. We extract features of MD trajectories using libraries PyEMMA [22] and MDTraj¹. Matrices are represented as NUMPY arrays and sparse matrices by scipy sparse matrices. We efficiently solve linear equations of the form $Ax = b$ using the PyPardiso 0.4.2² (version 2021.4) wrapper for the Intel oneMKL PARDISO library. For spectral clustering, we use the scikit-learn implementation. Our Python code for MOSCITO is publicly available³. For PCA, TICA and k-Means we use the implementations in PyEMMA. The implementation of SSC is based on [28] and a Python translation of SSC⁴. All experiments run on a virtual machine with 64 CPUs and 512GB of RAM running Ubuntu 22.04.

Datasets and parameter settings. We use MD trajectories of four proteins: 2F4K, Ace, 2WXC, and Savinase, which differ in the number of atoms $|A|$, number of $C\alpha$ atoms $|C\alpha|$, and number of residues $|R|$, as summarized in Table 1, together with number of trajectories and timesteps in each trajectory. If not specified otherwise, parameter default values are $d = 60$, $s = 3$, $\lambda_1 = 0.01$, $\lambda_2 = 15$, $a = 0.1$, $b = 0.1$.

¹<https://www.mdtraj.org>

²<https://github.com/haasad/PyPardisoProject>

³<https://github.com/Mhae98/MOSCITO>

⁴<https://github.com/abhinav4192/sparse-subspace-clustering-python>

⁵Birgit Schiøtt's Biomodelling group, Department of Chemistry, Aarhus University

Protein	A	C α	R	T	# Trajs	Source
2F4K	577	35	68	10.000	20	[25]
Ace	264	21	23	10.000	28	[11]
2WXC	710	47	47	10.000	10	5
Savinase	3.725	269	269	~1500	2	5

Table 1: Molecular dynamics data

6.1 MOSCITO settings and features

In this section, we analyze the parameter sensitivity and impact of feature extraction for MOSCITO. For the following experiments, 10 trajectories of the 2WAV-protein are used. The Markov state models are constructed for $k \in \{10, 50, 100, 200\}$ clusters with a lag time $\tau \in \{1, 10, 100\}$. The average VAMP2-score of the 10 trajectories is calculated for each choice of clusters k and lag time τ . The VAMP2-score is calculated for five eigenvalues. It is important to keep in mind that the VAMP2-score for different MSM lag times is not directly comparable, only for different numbers of clusters.

6.1.1 Dictionary size. d defines the dimensionality of the learned dictionary and thus the number of subspaces. As such, it is crucial for the clustering and runtime performance of MOSCITO, where larger dictionary size d provides more subspaces to represent the data. We study $d \in \{10, 20, \dots, 100\}$, and additionally $s \in \{2, 3, 4\}$ number of sequential neighbors. Figure 2 shows VAMP2-scores for different numbers of clusters (rows) and MSM lag times (columns): clustering performance improves for larger dictionary size d until it converges. For smaller MSM lag times, the VAMP2-score converges faster than for larger lag times, independent of the number of clusters. With an increasing number of sequential neighbors s , the overall shape of the curves is highly similar, with a slight preference for more neighbors that capture more of the temporal relationships.

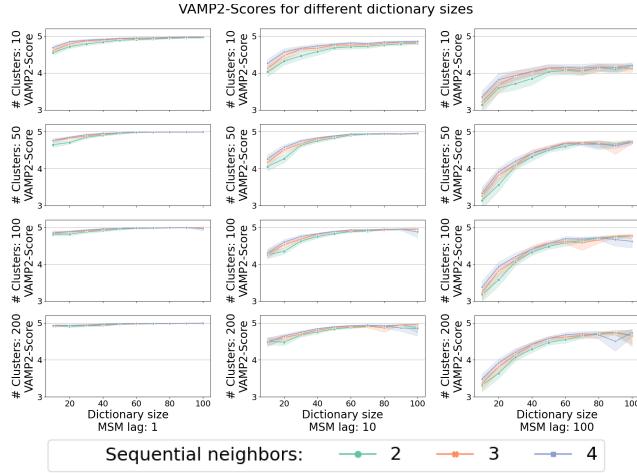


Figure 2: Comparison of the VAMP-scores for different dictionary sizes. For each combination of MSM lag and cluster count, the VAMP2-Score for different dictionary sizes and the number of sequential neighbors are compared. For better differentiation, the VAMP2-score is plotted starting at 3.

6.1.2 Sequential neighbors. Setting the number of sequential neighbors s in the temporal regularization to 0, the temporal aspect is not considered, with growing s longer temporal relationships are captured. The scores for $s \in \{0, \dots, 12\}$ are shown in Figure 3 for different MSM lags (columns) and number of clusters (rows). The number of sequential neighbors s clearly affects clustering performance, with too few or too neighbors reducing performance, but with a broad stable range where temporal relationships are successfully captured. Figure 4 shows the clustering of a 2F4K protein

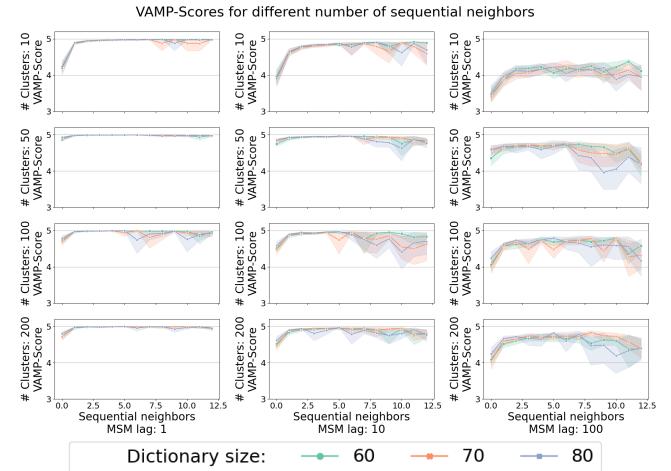


Figure 3: VAMP-scores for varying number of sequential neighbors and dictionary sizes over pairings of MSM lag and cluster count. For visibility, VAMP2-axis starts at 3. Best results for 3 to 5 sequential neighbors.

trajectory with a long folded state into 5 clusters represented by different colors. With no (top) or few (center) sequential neighbors, time intervals comprised by a cluster can be very short. Increasing the number of sequential neighbors (bottom) leads to continuous clusters with mostly uninterrupted blocks.

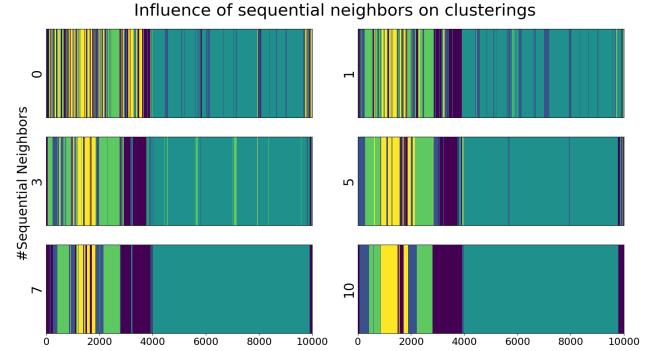


Figure 4: Varying number of sequential neighbors for 2F4K protein clustered into 5 clusters. Time steps are represented along the x-axis, clusters are implied by color.

6.1.3 λ parameters. We study settings of the tradeoff parameters $\lambda_1 \in \{0.01, 0.05, \dots, 15\}$ for sparsity of the coding matrix Z , $\lambda_2 \in \{0.01, 0.05, \dots, 35\}$ for the temporal regularization function. Figure 5 shows that best performance is achieved for low values of λ_1 and higher values of λ_2 (light areas in the heatmap). This shows that temporal regularization is the more important factor for optimal clustering performance.

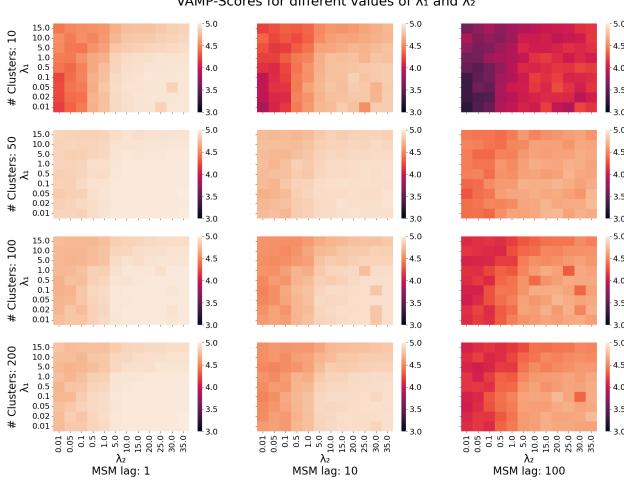


Figure 5: Heatmap of VAMP-scores (the higher/lighter, the better; scale from 3 to 5 for visibility); best results for values around 0.01 for λ_1 and around 15 for λ_2 .

6.1.4 Learning rates. $\alpha, \beta \in \{0.01, 0.05, \dots, 15\}$ define how much Lagrangian multipliers Π and Λ change in each iteration of the algorithm. Heat maps in Figure 6 show that the results are stable across parameter settings, with VAMP2-scores in each heatmap relatively close, meaning our default values work well.

6.1.5 Weighting mode. All weighting modes for the Laplacian regularization function: binary, Gaussian, logarithmic, and exponential, are tested for different numbers of sequential neighbors. In Figure 7 we see that for few sequential neighbors, the performance for all weighting modes is similar. For many sequential neighbors, performance starts to decrease. In most cases, the performance when using the binary weighting mode decreases earlier than for the other methods, but all reliably reach the same maximal score.

6.1.6 Feature comparison. We analyze the above described features and their combinations via simple concatenation. Figure 8 shows the average VAMP2-score for 10 trajectories of the 2WAX-protein, using the five largest eigenvalues for scoring.

Best results for individual features are achieved with backbone torsions. Overall, combinations of backbone torsions, SASA, and shape histograms yield the best results. Figure 9 shows the importance that MOSCITO gives to each of the features in its dictionary when clustering the full-dimensional feature space. While, e.g., the feature "flex" (flexible torsions) yields the worst VAMP2 score (left), it still has a high influence on the results (right), explaining the suboptimal VAMP2 score when using the full feature set.

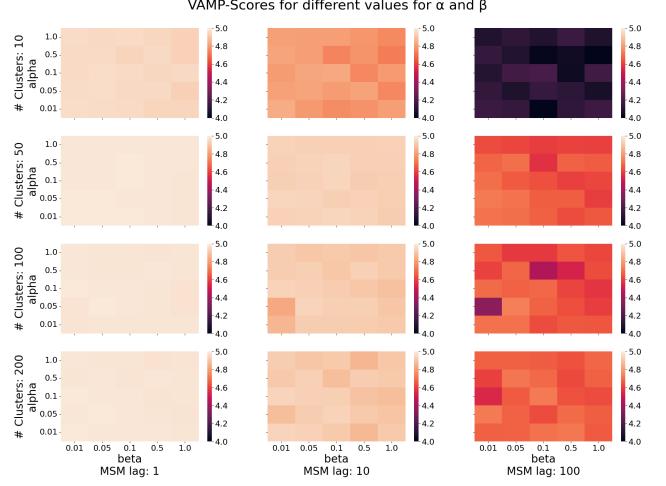


Figure 6: VAMP-scores for varying α and β : almost no impact on clustering performance.

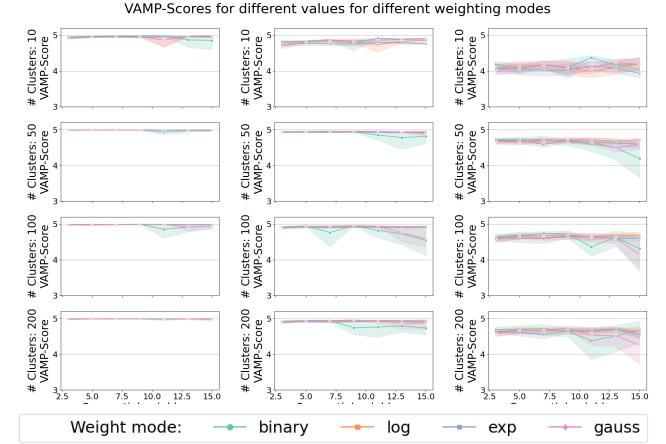


Figure 7: VAMP-scores for weighting modes of temporal Laplacian

6.2 Clustering performance

In this section, the clustering performance of MOSCITO is compared to that of state-of-the-art approaches that are currently used in the field. We compare to PCA with k -Means and to TICA with k -Means because of their widespread use [35]. Additionally, we compare against the sparse spectral clustering algorithm [28], a subspace clustering approach for molecular dynamics data. For PCA and TICA, the default parameter settings of PyEmma [35] were used. SSC is a subspace clustering algorithm, thus it does not need any dimensionality reduction in the preprocessing.

A general challenge in this kind of evaluation is the fact that no ground truth is given. As such, clustering performance is compared by constructing a MSM and scoring it using the VAMP2-score, in order to obtain quantitative insights. Still, visual inspection plays an

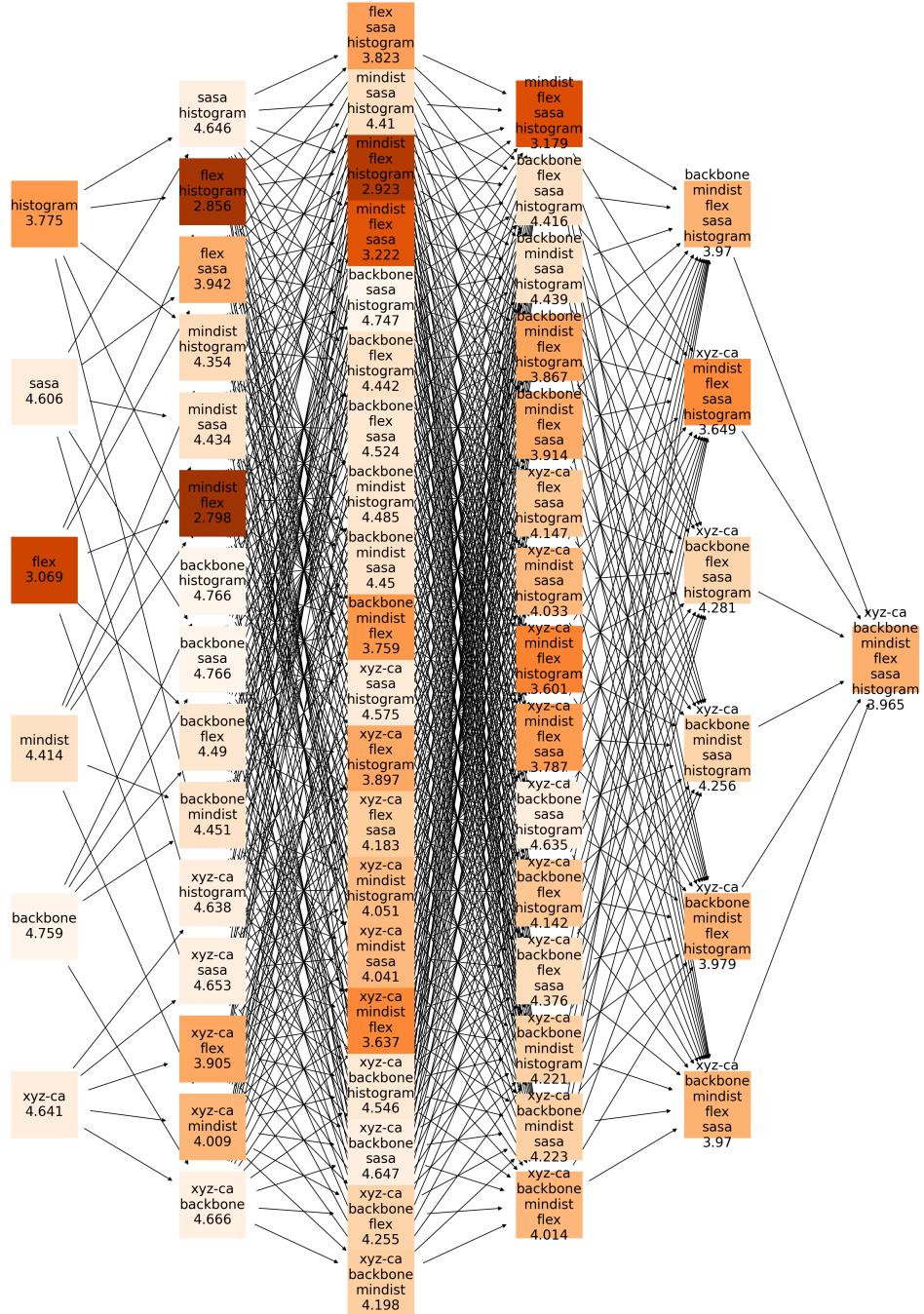


Figure 8: VAMP2-Scores for different featurizations (average over 10 trajectories for powerset of 6 features). Number of applied features increases from left to right. Colors correspond to the VAMP2-Scores, lighter values are better.

important role in assessing the clustering performance, in particular, the temporal connectivity of clusters.

Quantitative analysis. Figure 10 shows the average VAMP2-score for 20 trajectories of the 2F4K protein in the first row. For MSM lag time 1, the performance of all methods is comparable with the

exception of 10 clusters. There, MOSCITO and TICA + k -Means deliver the best performances. For longer MSM lag times 1 and 10, the performance for all methods is relatively similar for

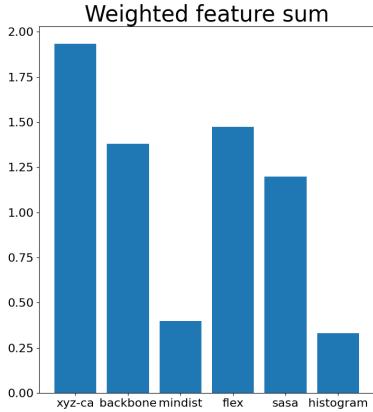


Figure 9: Weights (=relevance) of each feature according to the dictionary learned by MOSCITO when given the full set of features.

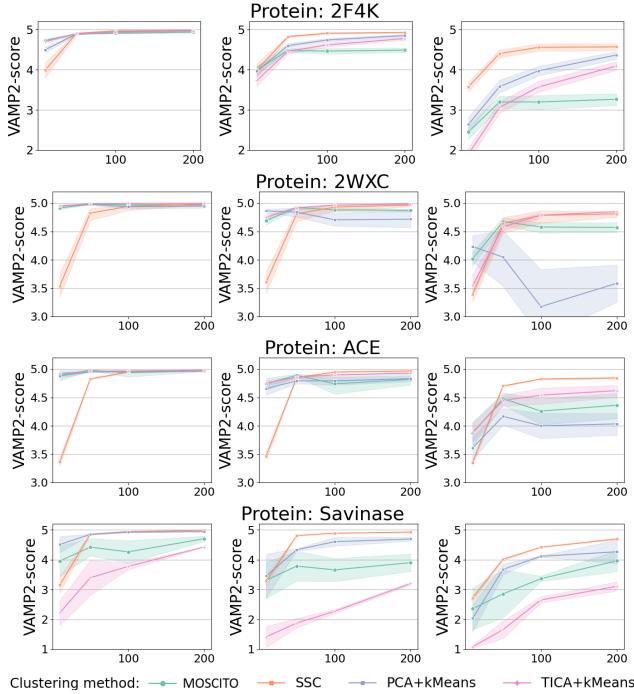


Figure 10: Mean VAMP2-Scores for 2F4K, 2WXC, ACE, and Savinase (top to bottom)

large cluster sizes. For small cluster sizes, SSC performs poorly, while the other methods already reach close to their maximal score for only 10 clusters. For MSM lag time of 100, PCA + k -Means drastically drops in performance, while the other methods maintain comparable scores to each other.

The third row of Figure 10 shows the average VAMP2-score for 28 trajectories of the Ace protein. Similar to the results of 2WXC, the performance for a MSM lag time of 1 and 10 is comparable between all methods for a large number of clusters. For few clusters, SSC

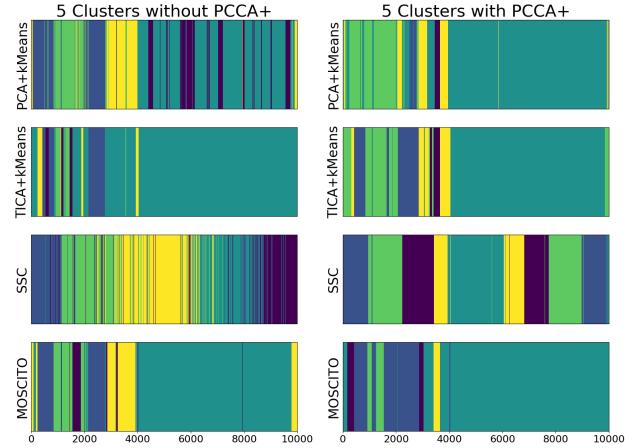


Figure 11: Clustering impact of PCCA+ on a 2F4K trajectory (each color represents a cluster; left: clustering the trajectory directly into 5 clusters; right: coarse-graining 50 clusters into 5 clusters using PCCA+).

again performs significantly worse. For MSM lag time of 100, the gap between the methods grows, with SSC performing best.

The last row of Figure 10 shows the average VAMP2-score for 28 trajectories of Savinase. For all MSM lag times, SSC performs best while TICA + k -Means performs worst. Unlike for the other proteins, all methods require a large number of clusters to reach the maximal VAMP2-score. This might be due to the large size of Savinase and shorter trajectories compared to the other proteins.

Qualitative analysis. The purpose of modeling temporal aspects is to obtain clusters with larger continuous segments, ideally retrieving macrostates directly. Figure 11 visually compares the segmentation of clustering methods, either by directly clustering the trajectory into five clusters or, alternatively by coarse-graining 50 clusters into five clusters using PCCA+ [20] (see Section 7). We regard a 2F4K trajectory with a known long folded segment.

Directly clustering, i.e. without PCCA+, PCA + k -Means and SSC struggle to find the large folded state as one large block. MOSCITO and TICA+ k -Means find similar clusterings and succeed to uncover the large folded state. When applying PCCA+, PCA + k -Means, TICA + k -Means, and MOSCITO generate similar clusterings, comparable to the ones of MOSCITO and TICA + k -Means without PCCA+. In this specific case, SSC fails to find the large segment, which is reflected in the respective VAMP2-score for this specific trajectory which is unusually low.

6.3 Clustering runtime

In this section, the runtime of MOSCITO is studied and compared to the runtime of PCA + k -Means, TICA + k -Means, and SSC. The runtimes for MOSCITO are measured for 20 iterations. The following runtimes are the average of 10 2WXC-trajectories clustered into ten clusters, using all atom coordinates as features. The main factors influencing the runtime of MOSCITO, besides the size of

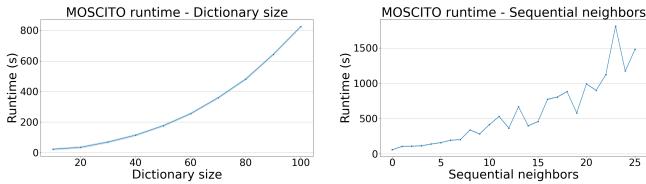


Figure 12: MOSCITO runtime (left: dictionary sizes; right: number of sequential neighbors).

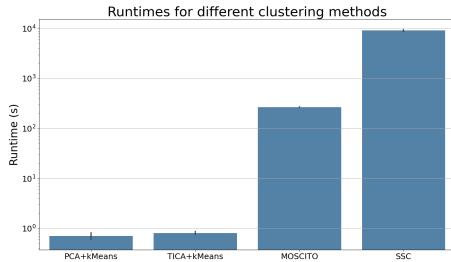


Figure 13: Runtime for PCA+k-Means, TICA+k-Means, MOSTCITO, SSC (logarithmic scale on y-axis for readability)

the input data, are the dictionary size d and the number of sequential neighbors s . The effect of different dictionary sizes [10, ..., 100] on the runtime is shown left in Figure 12. The effect of different numbers of sequential neighbors [0, ..., 25] is on the right.

The runtime of MOSCITO grows with increasing dictionary sizes. For optimal cluster performance, a large enough dictionary size has to be selected. Large values do not affect clustering performance negatively, but directly impact runtime. A dictionary size between 60 and 80 provides a good tradeoff. The runtime generally grows with increasing number of regarded sequential neighbors. The overall run-to-run variance for the different trajectories is close to zero, indicating a stable runtime for a fixed number of iterations.

For the comparison of the different clustering methods, backbone torsions were used. The comparison between the different clustering algorithms is shown in Figure 13. PCA + k -Means and TICA + k -Means are both much faster than the subspace clustering methods. Compared to SSC, MOSCITO is substantially faster.

6.4 Discussion

We conclude that MOSCITO provides at least comparable performance to state-of-the-art clustering methods for MD data. We evaluated the clustering performance by the clusters' suitability as Markov states using the VAMP2-score. The visual analysis of MOSCITO's clustering results shows a better clustering for higher numbers of sequential neighbors. Thus, MOSCITO is a very promising approach for creating MSMs in a one-step-approach.

MOSCITO's parameter robustness shows that there are good tradeoffs in dictionary size between runtime and clustering performance. The number of sequential neighbors matters, and our experiments indicate a good default value is between 3 and 5 to capture nearby temporal relations. In terms of feature extraction, MOSCITO on backbone torsions leads to best results, followed by

α coordinates and SASA. Combining features does not improve the overall clustering performance. While simpler non-subspace methods are even faster, MOSCITO provides a faster runtime than SSC, and state-of-the-art quantitative results, with qualitative analysis showing the promise of the approach.

7 RELATED WORK

Distance-based clustering algorithms suffer from the curse of dimensionality when it comes to high-dimensional data like MD data. Thus, the dimensionality is often reduced in a preprocessing step, e.g., with principal component analysis (PCA) or time-lagged independent component analysis (TICA) [12]. Both are widely used for analyzing MD data [15, 23]. PCA transforms the data onto the axes with the highest variance, also known as principal components. PCA is a commonly used for MD data, e.g., in PyEMMA [22], MSMBuilder⁶, or MDAAnalysis⁷. TICA (Time-lagged independent component analysis) [12] linearly projects the data to independent, uncorrelated components (IC). The ICs are computed by maximizing the autocovariance for a fixed lagtime τ , and are frequently used for MD data ([24], [17]).

State-of-the-art methods often follow a two-step approach, where the trajectories are discretized into a large number of *microstates*, and subsequently combined to the relevant *macrostates*. Since the transitions between the metastable states are taking place over multiple timesteps it is hard to assign time steps that are part of a transition to a metastable state clearly. Perron-Cluster Cluster Analysis (PCCA) ([34], [20]) and a more robust version PCCA+ ([33], [3]) fuzzily assign the microstates to their corresponding macrostates based on spectral methods for transition matrices:

The transition matrix P of a decomposable MSM with n_c macrostates can be reordered to a block-diagonal matrix \tilde{P} with n_c blocks. For the matrix X containing the n_c dominant eigenvectors of \tilde{P} , the rows belonging to states in the same block are identical [5]. These rows represent the n_c corners of a (n_c-1) dimensional simplex.

For MSMs with transition states, i.e., nearly decomposable MSMs, the structure of the simplex is only slightly perturbed. To assign the microstates to the macrostates, they are fuzzily assigned to the n_c corners of the simplex based on their position in the simplex. The goal is to find a linear transformation $A \in \mathbb{R}^{n_c \times n_c}$ so that $\chi = XA$ where χ_i are the membership vectors. Those memberships all have to be positive and sum up to 1. The resulting macrostates generally cannot be considered to be a Markov chain [20].

Time series data can be found in many fields, like traffic analysis, geology, computer vision, or molecular dynamics data. Many existing subspace clustering algorithms do not take the sequential properties of time series data into account [36]. One of the first subspace clustering methods taking advantage of sequential relationships in data was *SpatSC* introduced by [6]. A penalty term used to preserve sequential relationships in the learned affinity graph was added to the subspace clustering problem. Based on the same idea, *OSC*, a more robust subspace clustering method for sequential data was introduced by [29]. *TSC* [9] further improved on the idea by defining a Laplacian regularization to encode the temporal information. Our method MOSCITO builds upon *TSC*.

⁶<http://www.msmbuilder.org>

⁷<https://www.mdanalysis.org/>

To the best of our knowledge, only one subspace clustering method has been used on molecular dynamics data, namely SSC [28], which was first introduced in the field of image processing [4]. We include SSC in our experiments as comparative method, thus we introduce it in detail in the following. Note that SSC does not take advantage of any properties that distinguish time series from other high-dimensional data, leading to suboptimal use of the available information in MD data. The main idea behind SSC is to take advantage of the self-expressiveness of the data, i.e., assuming that every point $y_i \in Y$ can be rewritten as a linear combination of other points in the dataset. Each datapoint $y \in \cup_{i=1}^n S_i$ can be written as

$$y_i = Y c_i, \quad c_{ii} = 0, \quad (14)$$

where $Y = [y_1, \dots, y_N]$ and $c_i := [c_{i1} \ c_{i2} \ \dots \ c_{iN}]^T$ with the constraint $c_{ii} = 0$ eliminating the trivial solution of representing the point as itself. Since each subspace usually has more data points than dimensions the self-expressive dictionary Y is generally not unique. Among all those possible solutions for Equation 14 there is a sparse solution so that the non-zero entries of c_i correspond to datapoints in the same subspace as y_i . This solution can be found by minimizing the l_1 -norm of c_i , which can be solved efficiently and is known to prefer sparse solutions [4].

These sparse representations are transferred to an undirected graph where points that are part of another point's sparse representation are connected. Based on this graph, spectral clustering detects the desired subspace clusters.

8 CONCLUSION

We introduced MOSCITO, a subspace clustering method for high-dimensional molecular dynamics data. It uses temporal regularization to capture the relationship of sequential neighbors in time-series data. While traditional and state-of-the-art methods rely on coarse-graining using PCCA+ in a two-step approach, MOSCITO achieves similar overall performance in a single clustering step. Especially for small numbers of clusters, the temporal regularization in MOSCITO yields clusters consisting of connected time steps, fitting the real world more closely than alternative methods without relying on any postprocessing steps. Compared to current subspace clustering methods on MD data, MOSCITO provides a better runtime. In future work, the integration of a multi-view approach into MOSCITO might take advantage of the additional information provided by using multiple features.

REFERENCES

- [1] Mihai Ankerst, Gabi Kastenmüller, Hans-Peter Kriegel, and Thomas Seidl. 1999. 3D shape histograms for similarity search and classification in spatial databases. In *Advances in Spatial Databases: 6th International Symposium, SSD'99 Hong Kong, China, July 20–23, 1999 Proceedings*. Springer, 207–226.
- [2] Massimiliano Bonomi, Davide Branduardi, Giovanni Bussi, Carlo Camilloni, Davide Provasi, Paolo Raiteri, Davide Donadio, Fabrizio Marinelli, Fabio Pietrucci, Ricardo A. Broglia, and Michele Parrinello. 2009. PLUMED: A portable plugin for free-energy calculations with molecular dynamics. *Computer Physics Communications* 180, 10 (2009), 1961–1972. <https://doi.org/10.1016/j.cpc.2009.05.011>
- [3] Peter Deuflhard and Marcus Weber. 2005. Robust Perron cluster analysis in conformation dynamics. *Linear algebra and its applications* 398 (2005), 161–184.
- [4] Ehsan Elhamifar and René Vidal. 2013. Sparse Subspace Clustering: Algorithm, Theory, and Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 11 (2013), 2765–2781. <https://doi.org/10.1109/TPAMI.2013.57>
- [5] Anna-Simone Frank, Alexander Sikorski, and Susanna Röblitz. 2022. Spectral clustering of Markov chain transition matrices with complex eigenvalues. *J. Comput. Appl. Math.* (2022). under review.
- [6] Yi Guo, Junbin Gao, and Feng Li. 2014. Spatial subspace clustering for drill hole spectral data. *Journal of Applied Remote Sensing* 8, 1 (2014), 083644–083644.
- [7] F Ulrich Hartl. 2017. Protein misfolding diseases. *Annual review of biochemistry* 86 (2017), 21–26.
- [8] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. 2012. Subspace clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2, 4 (2012), 351–364.
- [9] Sheng Li, Kang Li, and Yun Fu. 2015. Temporal subspace clustering for human motion segmentation. In *Proceedings of the IEEE international conference on computer vision*. 4453–4461.
- [10] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. 2012. Robust recovery of subspace structures by low-rank representation. *IEEE transactions on pattern analysis and machine intelligence* 35, 1 (2012), 171–184.
- [11] Robert T. McGibbon. 2014. Fs MD Trajectories. (5 2014). <https://doi.org/10.6084/m9.figshare.1030363.v1>
- [12] Lutz Molgedey and Heinz Georg Schuster. 1994. Separation of a mixture of independent signals using time delayed correlations. *Physical review letters* 72, 23 (1994), 3634.
- [13] Frank Noé and Feliks Nuske. 2013. A variational approach to modeling slow processes in stochastic dynamical systems. *Multiscale Modeling & Simulation* 11, 2 (2013), 635–655.
- [14] Feliks Nuske, Bettina G Keller, Guillermo Pérez-Hernández, Antonia SJS Mey, and Frank Noé. 2014. Variational approach to molecular kinetics. *Journal of chemical theory and computation* 10, 4 (2014), 1739–1752.
- [15] Juliana Palma and Gustavo Pierdomini-Sottile. 2023. On the Uses of PCA to Characterise Molecular Dynamics Simulations of Biological Macromolecules: Basics and Tips for an Effective Use. *ChemPhysChem* 24, 2 (2023), e202200491. <https://doi.org/10.1002/cphc.202200491>
- [16] Lance Parsons, Ehtesham Haque, and Huan Liu. 2004. Subspace clustering for high dimensional data: a review. *Acm sigkdd explorations newsletter* 6, 1 (2004), 90–105.
- [17] Guillermo Pérez-Hernández, Fabian Paul, Toni Giorgino, Gianni De Fabritiis, and Frank Noé. 2013. Identification of slow molecular order parameters for Markov model construction. *The Journal of chemical physics* 139, 1 (2013), 07B604_1.
- [18] Wentao Qu, Xianchao Xiu, Huangye Chen, and Lingchen Kong. 2023. A Survey on High-Dimensional Subspace Clustering. *Mathematics* 11, 2 (Jan 2023), 436. <https://doi.org/10.3390/math11020436>
- [19] Julia Romanowska, Krzysztof S Nowinski, and Joanna Trylska. 2012. Determining geometrically stable domains in molecular conformation sets. *Journal of Chemical Theory and Computation* 8, 8 (2012), 2588–2599.
- [20] Susanna Röblitz and Marcus Weber. 2013. Fuzzy spectral clustering by PCCA+: Application to Markov state models and data classification. *Advances in Data Analysis and Classification* 7 (2013), 147–179. Issue 2. <https://doi.org/10.1007/s11634-013-0134-6>
- [21] Martin K. Scherer, Brooke E. Husic, Moritz Hoffmann, Fabian Paul, Hao Wu, and Frank Noé. 2019. Variational selection of features for molecular kinetics. *The Journal of Chemical Physics* 150, 19 (may 2019), 194108. <https://doi.org/10.1063/1.5083040>
- [22] Martin K. Scherer, Benjamin Trendelkamp-Schroer, Fabian Paul, Guillermo Pérez-Hernández, Moritz Hoffmann, Nuria Plattner, Christoph Wehmeyer, Jan-Hendrik Prinz, and Frank Noé. 2015. PyEMMA 2: A Software Package for Estimation, Validation, and Analysis of Markov Models. *Journal of Chemical Theory and Computation* 11 (Oct. 2015), 5525–5542. <https://doi.org/10.1021/acs.jctc.5b00743>
- [23] Steffen Schultze and Helmut Grubmüller. 2021. Time-Lagged Independent Component Analysis of Random Walks and Protein Dynamics. *Journal of Chemical Theory and Computation* 17, 9 (2021), 5766–5776. <https://doi.org/10.1021/acs.jctc.1c00273> PMID: 34449229.
- [24] Christian R Schwantes and Vijay S Pande. 2013. Improvements in Markov state model construction reveal many non-native interactions in the folding of NTL9. *Journal of chemical theory and computation* 9, 4 (2013), 2000–2009.
- [25] David E. Shaw, J.P. Grossman, Joseph A. Bank, Brannon Batson, J. Adam Butts, Jack C. Chao, Martin M. Deneroff, Ron O. Dror, Amos Even, Christopher H. Fenton, et al. 2014. Anton 2: Raising the Bar for Performance and Programmability in a Special-Purpose Molecular Dynamics Supercomputer. In *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 41–53. <https://doi.org/10.1109/SC.2014.9>
- [26] A. Shrike and J.A. Rupley. 1973. Environment and exposure to solvent of protein atoms. Lysozyme and insulin. *Journal of Molecular Biology* 79, 2 (1973), 351–371. [https://doi.org/10.1016/0022-2836\(73\)90011-9](https://doi.org/10.1016/0022-2836(73)90011-9)
- [27] Ernesto Suárez, Rafal P. Wiewiora, Chris Wehmeyer, Frank Noé, John D. Chodera, and Daniel M. Zuckerman. 2021. What Markov State Models Can and Cannot Do: Correlation versus Path-Based Observables in Protein-Folding Models. *Journal of Chemical Theory and Computation* 17 (5 2021), 3119–3133. Issue 5. <https://doi.org/10.1021/acs.jctc.0c01154>
- [28] Ivan Syzonenko and Joshua L. Phillips. 2018. Hybrid Spectral/Subspace Clustering of Molecular Dynamics Simulations. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*

- (Washington, DC, USA) (BCB '18). Association for Computing Machinery, New York, NY, USA, 325–330. <https://doi.org/10.1145/3233547.3233595>
- [29] Stephen Tierney, Junbin Gao, and Yi Guo. 2014. Subspace clustering for sequential data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1019–1026.
 - [30] Ivana Tošić and Pascal Frossard. 2011. Dictionary learning. *IEEE Signal Processing Magazine* 28, 2 (2011), 27–38.
 - [31] Nagarajan Vaidehi and Abhinandan Jain. 2015. Internal coordinate molecular dynamics: A foundation for multiscale dynamics. *The Journal of Physical Chemistry B* 119, 4 (2015), 1233–1242.
 - [32] René Vidal. 2011. Subspace clustering. *IEEE Signal Processing Magazine* 28, 2 (2011), 52–68.
 - [33] Marcus Weber and Tobias Galliat. 2002. Characterization of transition states in conformational dynamics using Fuzzy sets. (2002).
 - [34] Marcus Weber, Wasinee Rungsarityotin, and Alexander Schliep. 2004. *Perron cluster analysis and its connection to graph partitioning for noisy data*. Citeseer.
 - [35] Christoph Wehmeyer, Martin K. Scherer, Tim Hempel, Brooke E. Husic, Simon Olsson, and Frank Noé. 2019. Introduction to Markov state modeling with the PyEMMA software [Article v1.0]. *Living Journal of Computational Molecular Science* 1 (2019). Issue 1. <https://doi.org/10.33011/livecoms.1.1.5965>
 - [36] Fei Wu, Yongli Hu, Junbin Gao, Yanfeng Sun, and Baocai Yin. 2015. Ordered subspace clustering with block-diagonal priors. *IEEE transactions on cybernetics* 46, 12 (2015), 3209–3219.
 - [37] Hao Wu and Frank Noé. 2017. Variational approach for learning Markov processes from time series data. <https://doi.org/10.48550/ARXIV.1707.04659>