A Visual Analytics Workflow for Detecting Transition Regions in Large Scale Molecular Dynamics Simulations

by

Rostyslav Hnatyshyn

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved June 2022 by the
Graduate Supervisory Committee:

Ross Maciejewski, Chair
Chris Bryan
James Ahrens

ARIZONA STATE UNIVERSITY

August 2022

ABSTRACT

Molecular Dynamics (MD) simulations are ubiquitous throughout the physical sciences; they are critical in understanding how particle structures evolve over time given a particular energy function. A software package called ParSplice introduced a new method to generate these simulations in parallel that has significantly inflated their length. Typically, simulations are short discrete Markov chains, only capturing a few microseconds of a particle's behavior and containing tens of thousands of transitions between states; in contrast, a typical ParSplice simulation can be as long as a few milliseconds, containing tens of millions of transitions. Naturally, sifting through data of this size is impossible by hand, and there are a number of visualization systems that provide comprehensive and intuitive analyses of particle structures throughout MD simulations. However, no visual analytics systems have been built that can manage the simulations that ParSplice produces. To analyze these large data-sets, I built a visual analytics system that provides multiple coordinated views that simultaneously describe the data temporally, within its structural context, and based on its properties. The system provides fluid and powerful user interactions regardless of the size of the data, allowing the user to drill down into the data-set to get detailed insights, as well as run and save various calculations, most notably the Nudged Elastic Band method. The system also allows the comparison of multiple trajectories, revealing more information about the general behavior of particles at different temperatures, energy states etc.

ACKNOWLEDGMENTS

I gratefully acknowledge that this material is based upon work supported by the U.S. Department of Energy, Office of Science, and the National Security Education Center (NSEC) through the Exascale Computing Project. Additionally, this material is based upon work supported by the U.S. Department of Homeland Security under Grant Award Number 17STQAC00001. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

I would also like to thank my committee members for their guidance and support throughout this project, and for getting me acquainted with all of the immensely talented staff at Los Alamos National Laboratory, for without them, this project would not even exist. I would also like to thank Danny Perez for his immense patience in tutoring me about physics and molecular dynamics, as well as taking the time to provide constant constructive feedback on the system. I would also like to thank Divya Banesh and Jieqiong Zhao for mentoring me throughout the process, and showing me what it means to do cutting-edge research.

TABLE OF CONTENTS

CHAPTER Page

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

INTRODUCTION

Molecular dynamics (MD) simulations enable scientists to observe how particles evolve over time assuming a potential energy function. Each step of a MD simulation is typically an extraordinarily small length of time (e.g, femtoseconds; 10e-15); this time-scale is chosen based on the fastest vibrational frequency in the system - choosing time-scales larger than the system's vibrational frequencies can lead to missing important transitions between states, as these transitions depend on the particle vibrating. MD simulations have found applications in drug design (Hospital *et al.*, 2015; Durrant and McCammon, 2011), biophysics (Berendsen, 1987), materials science (Miyazaki and Shiozaki, 1996), and numerous other fields in the physical sciences. A large family of software packages have been developed in order to generate these simulations, such as GROMACS (Bekker *et al.*, 1993) for biological simulations, LAMMPS (Thompson *et al.*, 2022) for materials modeling, as well as countless others in other fields. These software packages simulate the interactions of atoms within the particle by using Newton's laws of motion. A simulation management system called ParSplice (Perez *et al.*, 2016) was recently introduced, which enabled MD simulations to span relatively long time-scales, provided that the particles being simulated are of a modest size.

MD simulations generate thousands of transitions between discrete particle structures, referred to as *states*. States contain meta-data about the particle being simulated at a given point in time, such as its position, species, etc. These states tend to repeat numerous times throughout a simulation, often cycling through a set of neighbor states rapidly.

A chain of states is called a *trajectory*, and a single trajectory describes one of the many possible ways a particle can behave given a potential energy function. Trajectories generated by ParSplice are mostly comprised of *energy basins* (Figure 1.1), long periods of time where the simulation oscillates between a small subset of states, remaining structurally similar throughout. This is caused by the fact that transitioning between these states does not require a large amount of energy, but transitioning to another basin (set of states) does. Regions where the atoms drastically change their structure and overcome the energy barrier between basins occur rarely and usually within the span of a few timesteps (Henkelman, 2017). These regions are called *transition regions* (Figure 1.1), and the heterogeneity of energy barriers has been a long-standing problem in simulating long MD trajectories.

Despite the obvious scientific value of these simulations, trajectories with the heterogeneous energy barrier problem are difficult to analyze due to the large amounts

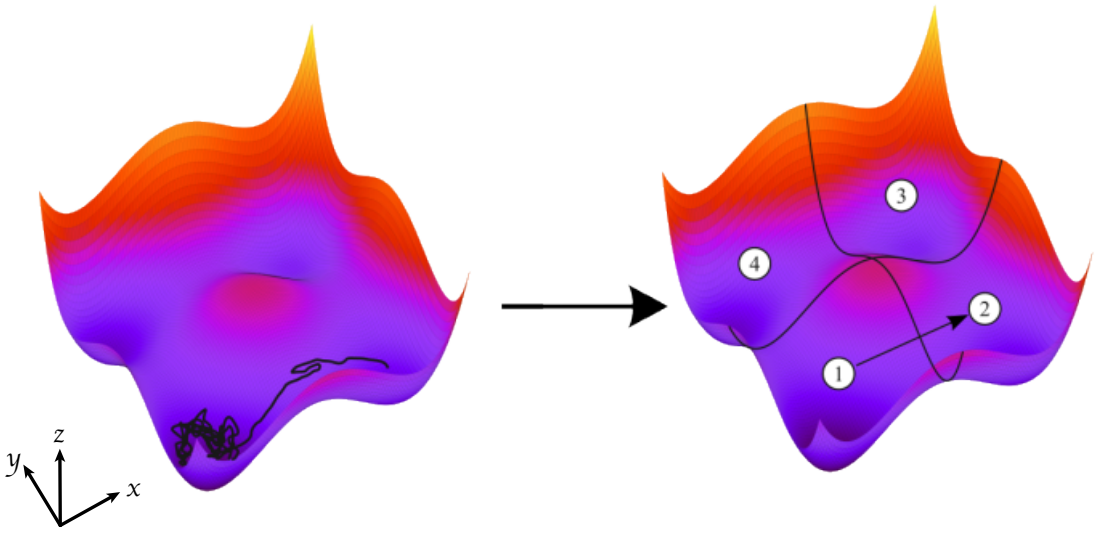

Figure 1.1: A 3D Surface Demonstrates the Energy (Z-axis) of a Particle Throughout a Simulation. The Black Line (Left) Illustrates the Trajectory the Particle Has Taken. The Numbers (Right) Indicate Hypothetical *Energy Basins* Which Usually Correspond to Different Structures of a Particle. *Transition Regions* Can Be Regarded as Lines Separating These Basin Regions. Figure Was Reproduced with Permission from the Original Author, Danny Perez (danny_perez@lanl.gov)).

of superfluous data they produce. Since there are a large amount of uninteresting transitions (i.e energy basins) even within a small data-set, there is a need to develop a compact, progressive visualization (Fekete *et al.*, 2019) that highlights the essential components of the trajectory (i.e., the transition regions), while understating the parts of the trajectory where there is little to no change in the structure of the particle. To contribute to the problem further, simulations generated by ParSplice contain states that transition to their topologically equivalent "symmetric" cousins that differ only in their atomic labeling, yielding no new insights yet making analysis more difficult. Such a workflow was explored in Huang *et al.* (2017, 2018) but has not been automated nor visualized.

A large variety of visual analytics systems exist for molecular dynamics simulations (Mixcoha *et al.*, 2016; van der Kamp *et al.*, 2010; Skånberg *et al.*, 2018; Newport *et al.*, 2019; Chae *et al.*, 2019; Rodríguez-Espigares *et al.*, 2020), however, none have been built specifically for the analysis of transitions between energy basins, especially for the kinds of trajectories that are generated by ParSplice. The data-sets explored in this work are long-duration, heterogeneous energy barrier trajectories that were generated through the use of accelerated molecular dynamics techniques (AMD); due to the relative novelty of these techniques, few visual analytics tools have been developed for them as of the time of this writing.

Ultimately, this work facilitates the exploratory analysis of these trajectories by providing an aesthetically pleasing and user-friendly interface through a web-application, as well as providing a back-end library for domain experts to efficiently access a graph database containing simulation data. Analyzing an ensemble of trajectories can explain a particle's behavior at different temperatures and energy states, which can be used to qualify a novel material's usefulness and robustness to extreme conditions.

This work consists of three major contributions:

- A visual analytics system that seamlessly integrates a overview of trajectory ensembles, a detailed view of node-link diagrams as well as a series of supplementary coordinated views that support the interactive analysis of transitions between energy basins;
- A data processing library that manages large-scale molecular dynamics data and supports the automatic detection of transition regions;
- Two case studies of the system where a domain expert identifies and compares regions of interest within an ensemble of trajectories.

The contents of the rest of this work are as follows: chapter 2 covers related scientific background information; chapter 3 focuses on related work; chapter 4 covers the system's architecture as well as its individual components; chapter 5 covers a case study of platinum nano-particles; chapter 6 discusses the results of the case study, as well as the limitations and benefits of the system that were revealed by the case study of the system; finally, chapter 7 concludes the thesis and discusses future work.

Chapter 2

BACKGROUND

In this chapter, relevant scientific background will be covered. This chapter explains the mechanics of molecular dynamics simulations, and then covers how simulations generated by ParSplice differ from traditional MD techniques. Then, the heterogeneous barrier problem is covered, and understanding it is key to grasping what my system is attempting to accomplish. Following the description of the heterogeneous barrier problem is a description of nano-particles, as well as their structures. Finally, the chapter concludes with explanations of established methods used to explore transition regions that are incorporated into the system.

2.1 Molecular Dynamics Simulations

Molecular dynamics (MD) simulations use Newton's law of motion to explain the detailed movements of atoms alone as well as within molecules/particles. Traditional MD simulations use Newton's laws of motion to evolve the system starting from an initial state; while this works well for simple systems, there are certain classes of systems that cannot be easily simulated due to their prohibitive computational costs. This primarily occurs with long-duration simulations, as they are typically made up of energy basins where the trajectory states in a small sub-set of states for a long time. Traditionally, Transition State Theory (TST) (Truhlar *et al.*, 1996) was used to quantitatively explain chemical reactions, but increasing interest in high-complexity systems made this approach impractical; this is due to the fact that complex systems tend to have large sets of possible paths the reaction can take. This weakness inherent in TST caused scientists to look elsewhere to explain chemical reactions.

Accelerated molecular dynamics (AMD) techniques (Hamelberg *et al.*, 2004) allow the simulation of systems that were previously thought to be impossible to simulate through the exploitation of the mathematical properties of MD simulations. The Parallel Replica technique (Voter, 1998) generates N copies of a simulation at each time-step and runs them until one of the simulations transitions to another state. Hyper-dynamics (Voter, 1997) adds a potential energy bias to each state so that the simulation overcomes the energy barrier between states faster. Another technique is to simply run simulations at higher temperatures, decreasing their reaction times, and then filtering out transitions that would not occur at lower temperatures. This list is far from conclusive, as there is a wealth of other AMD techniques available that are used for other molecular systems. Despite the prevalence of alternative techniques to run MD simulations, they still fall short of solving the **heterogeneous barrier problem**.

2.2 Heterogeneous Barrier Problem

There are certain classes of systems where the energy barriers are not uniform; i.e, some transitions require a high amount of energy to occur, while other transitions in the system require low energies; this is known as the heterogeneous or low barrier problem (Miron and Fichthorn, 2004). This problem is especially pronounced in systems where the molecules rapidly bounce around in one set of states for a long time before making a slow, drastic change to another set of states. These low energy barriers (fast transitions) can be easily simulated with standard MD techniques, while high energy barriers are addressed by AMD techniques. However, in this case, using AMD techniques would fail because they are only sped up by their fastest processes. In other words, the low-energy barrier transitions would be the only ones to benefit from using AMD techniques, as they take into account only the fastest processes.

The lack of a solution to this problem left a number of interesting systems difficult to generate until ParSplice was introduced.

2.3 ParSplice

ParSplice was developed with the heterogeneous barrier problem in mind, and uses the power of both standard and accelerated MD techniques to generate simulations. This implies that ParSplice is not a classic simulation software; rather, it is a simulation manager that takes results from multiple simulations and compounds the results to generate data that otherwise could not be generated.

The way ParSplice works is partially inspired by the Parallel Replica Dynamics (ParRep) technique (Voter, 1998), and Parsplice gains most of its computational efficiency through the clever use of parallelism. ParSplice works by simulating the all state-to-state transitions separately in parallel, and then splicing them together; standard MD techniques can be used for the low-energy barrier transitions, while AMD techniques can be used to simulate the high-energy transitions. Segments of the trajectory that start and end in the same state can be spliced to other segments and are simulated in parallel - this is in contrast to ParRep which runs N simulations at a time in hopes of finding the transition to the next state. ParSplice uses statistical methods to predict what segments are needed in the trajectory, and then splices them together in order. After the simulation is spliced together, these segments are discretized into a Markov State Model (MSM) and placed into one long chain that is described in the trajectory file generated by ParSplice. This reduces the size of trajectories significantly as it compresses the trajectory file into transitions between states instead of focusing on each individual time-step. If this compression method was not used, trajectory files would be infeasible to read and write. For instance, a millisecond long simulation would contain 10^{12} lines (10^{12} femtoseconds in

a millisecond). Each line would on average take 14 bytes, assuming that each state ID was 10 characters long (10 bytes), each duration was on average 3 characters long (3 bytes), and including the necessary 0 (1 byte) at the end of the line. Not including symmetry lines (adding more lines each 3 bytes long), an uncompressed file of such length would be $1.4 * 10^{14}$ bytes, or 14 terabytes.

2.4 Canonical States

A factor that makes it difficult to analyze raw MD simulation data from ParSplice is the presence of *canonical* and *non-canonical* states. These states are topologically equivalent to each other, and are marked as such during the simulation. While the simulation is running, non-canonical states are switched to their canonical representation in order to save some computational effort, as restarting the simulation from a new state is slower than working from a previously explored state. However, this comes with a few disadvantages, particularly when handling the data - the canonical states should be used for visualization purposes, as well as in computations such as the Perron-Cluster Cluster Analysis (PCCA) (Deuflhard and Weber, 2005), but the non-canonical versions must be used while using Nudged Elastic Band (NEB) (Jónsson *et al.*, 1998) methods, covered in detail further in this section.

Non-canonical states must be used for NEBs because NEBs can only be performed between actual state transitions, and not when states are remapped to their canonical versions. To further complicate things, NEBs cannot be performed between states that do not belong to the same segment when generated by ParSplice; this is because NEBs rely on the fact that atom labels match up exactly, and there is no guarantee atom labels will match up between a state within a segment and a canonical representative.

2.5 Nano-particles

Nano-particles are defined as particles of matter between 1 and 100 nanometers (nm) in diameter (Vert *et al.*, 2012) - far smaller than what is visible to humans. The main interest in studying nano-particles is the fact that their properties are wildly different in relation to their aggregate form.

These changes are caused primarily by the fact that the ratio of surface area to volume is increased, leading to the surface of each particle becoming the center of activity. (Sajanlal *et al.*, 2011) Moreover, nano-particles are small enough to be subject to various quantum effects, further exaggerating their differences from their aggregate form. Quantum effects are highly influential at the nano-scale precisely because of the size of the particles; the electrons within each atom in the particle behave differently due to the restricted amount of space they have to move around in. (Roduner, 2006) As such, it is difficult to predict the properties of nano-particles without directly observing them, thus dictating the need to generate their trajectories with advanced MD simulation techniques. Certain metal nano-particle simulations are characterized by their heterogeneous energy barriers, making ParSplice a natural choice for simulation.

Platinum nano-particles are particularly interesting because they have a non-spherical shape and belong to the family of metal nano-particles; metal nano-particles tend to have applications in catalyzing chemical reactions and building optical devices. (Stepanov *et al.*, 2014) Platinum nano-particles in particular have been used to reduce pollution from cars (Bell, 2003), develop new kinds of fuel cells (Balbuena *et al.*, 2010), and manufacture various electronic parts (Yoshida *et al.*, 2009).

2.6 Crystalline Structures

In order to produce a meaningful analysis of a MD simulation, an analyst needs to identify the characteristic structures of the trajectory, as well as the structures along transition paths. The characteristic structures of a trajectory are often derived using various clustering algorithms that attempt to group together similar states. Characteristic structures are often the ones the simulation spends the most time oscillating to, and they describe the essence of the reaction being studied. Structures along transition paths reveal exactly how the particle changed its form over the course of a reaction, allowing domain experts to qualify exactly how the reaction took place.

The most common crystalline structures found in metals are the face-centered cubic structure (FCC), the hexagonal close-packed structure (HCP), the body-centered cubic structure (BCC), and the icosahedral coordination structure (ICO). These structures are most common in metals because the atoms within the particles tend to pack together in order to fill space as densely and efficiently as possible. All of these structures are differentiated by the relationships between the angles between sides of the unit cell, which can be thought of as a bounding box, and the distance between points in the cell.

These structures can be quantified within a particle's structure through Ackland-Jones analysis (Ackland and Jones, 2006) and the Common Neighbor Analysis (CNA) (Faken and Jónsson, 1994). The Ackland-Jones analysis uses the angles between pairs of atoms to derive how many of the atoms within the particular structure conform to one of the above-mentioned structures. Meanwhile, CNA works by first determining the structures of all of the atoms in the set, and then classifying each atom by the structures of its nearest neighbors.

2.7 Perron-Cluster Cluster Analysis

Perron-Cluster Cluster Analysis (PCCA) (Deuflhard and Weber, 2005) has been demonstrated to produce meaningful metastable sets (clusters) of MD trajectories based on the kinetic properties of a trajectory (Huang *et al.*, 2017; Cordes *et al.*, 2002). PCCA is commonly used in biological MD trajectories, but has been recently been demonstrated to provide insight into the behavior of nano-particles (Huang *et al.*, 2018). PCCA works by calculating the eigenvectors of a trajectory's transition matrix and uses them to build meta-stable sets of states. These meta-stable sets are groups of states that represent energy basins within a trajectory; PCCA clustering can be used to highlight when the trajectory shifts from energy basin to energy basin. It should be noted that PCCA only takes into account the transitions between states, and not their actual molecular structure. For this reason, PCCA can be used as a tool to determine where to look for structural changes, but it does not identify them entirely - manual analysis is still needed to determine where exactly a transition region occurs.

The GPCCA algorithm (Reuter *et al.*, 2018), used in NeoMD, uses Schur decomposition (Schott, 2016) in order to speed up the computation of the PCCA, and is robust enough to work on simulations that do not have thermodynamic equilibrium, which opens up the analysis to a wide array of systems, including the system of platinum nano-particles explored in the case study (chapter 5).

2.8 Nudged Elastic Band

The goal of the Nudged Elastic Band (NEB) calculation is to find the minimum energy path (MEP) between temporally adjacent states. In order to find the MEP, an optimization problem must be solved over a 3D surface called the potential energy

surface (PES), which describes the potential energy for each atom in the system in terms of their positions. Saddle points within this energy surface correspond to transition states, and between any two minima, the MEP will necessarily pass through a maximum at a saddle point, hence the need to calculate the MEP between states. Minima themselves correspond to physically stable states - in other words, the states that were chosen for the calculation themselves. Finding the saddle point along the MEP provides the activation energy barrier, the quantity of energy that is needed to transition from one state to another. Deriving the activation energy barrier quantity is necessary to estimate the transition rate between two states.

The NEB itself is an optimization problem that tries to find the saddle-points of the PES along the MEP. NEB implementations typically allow the user to set an arbitrary number of intermediate images between states on the path; these images aim to find the lowest energy position possible while maintaining equal distance between their neighbors. By optimizing intermediate images, the energy curve gets smoothed out and the energy trajectory between two states becomes clearer.

Chapter 3

RELATED WORK

Visual analytics systems for exploring large data-sets have been keeping up with the current big data explosion (Naeem *et al.*, 2022) that has been occurring over the past decade - many important decisions made by businesses, corporations and governments throughout the world are heavily reliant on the use of analytical tools that can produce valuable insights on large data-sets. MD simulation visual analytics (VA) systems are no different, and there have been a number of VA systems that explore long-duration MD simulations (Chae *et al.*, 2019; Hildebrand *et al.*, 2019; Skånberg *et al.*, 2018). However, the incredibly long duration trajectories that can be handled by NeoMD (proposed in this thesis) have not been explored previously because they were not available prior to the introduction of ParSplice. NeoMDWeb (the web application proposed in this thesis) was designed with these long duration data-sets in mind, prioritizing optimized and fluid views over computationally complex visualizations. Moreover, the focus of the system is to highlight basin-to-basin transitions, instead of state-to-state transitions; this is what primarily sets it apart from other MD visual analytics systems. Additionally, most MD VA systems focus on biological macro-molecules - such visualization and analysis methods are not suited for the small nano-particle simulations generated by ParSplice.

Chae *et al.* (2019) explored long duration MD simulations in the context of biology. They used dimensionality reduction techniques to simplify the data-set before visualizing it in a 3-Dimensional context. The authors also admit that the interactions to select clusters within a 3-Dimensional context are still clumsy; as selection interactions are typically done in 2-Dimensions. My approach to overcome the curse

of dimensionality lies in focusing purely on the relationships between states, something made possible by Neo4j. In my system, the PCCA algorithm and sequence view are rendered simply by placing the database ID of each state in a sequence and using that ID to index into an array containing the relevant metadata. Through keeping the data representation as simple as possible, my system allows direct access to data relevant to the user's interest without losing any of its fidelity caused by using machine learning techniques utilized in other systems.

There also exist a number of analytical libraries that can process MD data (McGibbon *et al.*, 2015; Michaud-Agrawal *et al.*, 2011; Roe and Cheatham, 2013; Seeber *et al.*, 2007). However, these libraries, much like the currently available visual analytics systems, focus on MD systems that do not suffer from the heterogeneous barrier problem, or they are specifically built for the analysis of bio-molecular systems. MDTraj (McGibbon *et al.*, 2015) serves more as a bridge for bringing obscure data formats to known analysis tools, but its low-level interface proves to be challenging for analysts to use. In the future, NeoMD may come to leverage some of the computations made possible by these analytical libraries.

Visual analytics systems for analyzing spatio-temporal problems have been explored in Steptoe *et al.* (2018); Tominski *et al.* (2012) and more (Paula Afonso *et al.*, 2020; Liu *et al.*, 2019; Andrienko and Andrienko, 2021). The pixel-based representation of a trajectory proposed in Steptoe *et al.* (2018) was adopted within my system, with major changes, since my system usually only deals with a limited number of trajectory sequences at one time, and these sequences are very large compared to the ones explored in the paper.

The stacking trajectory approach proposed in Tominski *et al.* (2012) suggests using colors for encoding attribute values, but applying this to the sequence view would result in the PCCA clustering context being lost; moreover, encoding attributes

through color has a number of weaknesses, including diminishing accessibility to color-blind persons. Each state has at least a dozen "default" attributes generated by DBExtract (proposed in this thesis), not to mention the attributes that are calculated in various analyses provided by the system. Encoding these values with color would result in a loss of context for the properties. Instead, I decided to use a coordinated scatter-plot view alongside the other views to allow the user to compare portions of the sequence and its properties.

The immense size of the trajectories that my system visualizes has led to the development of a stacked bar-chart approach that uses semantic zooming techniques (Buering *et al.*, 2006; Conti *et al.*, 2005; Dunsmuir, 2011; Garcia *et al.*, 2011), a well-established approach for displaying large amounts of data in limited screen space. Semantic zooming allows the system to hide extraneous data from the user in order to reduce both the cognitive and computational load of the data-set. The semantic zooming available in the sequence view is coordinated with the graph view.

A highly scalable graph view was proposed in Abello *et al.* (2006); the authors used clustering algorithms to build hierarchies on arbitrary data-sets in order to enable the exploration of very large data-sets. In their system, the hierarchies are explored through clicking on the nodes that the user is interested in a tree-like interface. This approach is similar to the one employed in my system - I used PCCA to generate a clustering for the data-set, and then simplified the results into hierarchies based on the temporal occurrence of each state that was deemed important. Then, through zooming on the sequence view, the user explores the hierarchy of the clustering - the graph view updates dynamically according to the view-port of the sequence view. The sequence view itself is an implicit tree-view, as there is only one level that changes based on the users zoom level; the user is not interested in the hierarchy between levels, only the difference between a group of states and their individual data points.

My system has multiple-coordinated views (Roberts, 2007), providing the user with multiple contexts to explore the data in - the graph focuses on relations, the sequence provides temporal context, and the scatter-plot view provides detail. This allows for a multi-dimensional look at the data that encompasses all of the possible relationships between a state's neighbors, their temporal significance and their structural properties.

The scatter-plot grid displayed on the main view was inspired by Elmqvist *et al.* (2008). My scatter-plot grid is admittedly much simpler than the one discussed - it does not support any of the inter-scatter-plot interactions that are covered in the paper; moreover, my system does not support the automatic creation of scatter-plots based on available attributes due to computational constraints, as well as lack of screen real-estate. However, the coordination between the scatter-plot matrix and the other relational and temporal views still makes it a valuable component in the system.

Craig and Kennedy (2003) discuss a coordinated scatter-plot similar to the ones used in the scatter-plot matrix; the work discusses the weaknesses of overviews displaying detailed properties about the data, and uses a similar approach to highlight the properties of a data-set. However, my views are coordinated in multiple ways, and the work does not consider saving "snapshots" of user selections the way my system does with the sub-sequence view.

I applied concepts from common trajectory mining techniques (Uddin *et al.*, 2011; Mazimpaka and Timpf, 2016) in order to identify regions of interest within MD trajectories. In my system, PCCA gets applied before the trajectory is visualized; I used the fuzzy memberships generated by PCCA as criteria for whether or not a state should be simplified or presented to the user. As discussed in the background section, PCCA is a coarse-grained method that reveals kinetic relationships between states;

this was the main driving force behind adopting it as our method for finding regions of interest. The simplification algorithm in NeoMD functions primarily through outlier detection; it searches through the list of unique states to find states that have a membership probability below a user-defined threshold and places them into chunks that can be explored, hiding the rest of the unimportant information.

Chapter 4

SYSTEM

My proposed system is a computational pipeline composed of three major components: DBExtract, NeoMD, and the front-end interface, called NeoMDWeb. A sample workflow is provided in Figure 4.1, NeoMDWeb focuses on simulation data generated by ParSplice, particularly in systems where the energy barrier are heterogeneous. However, the system can work with any kind of simulation data, provided that it is represented as a discrete Markov chain. This chapter will describe the intent behind the system, as well as go over the major features of the user interface and back-end.

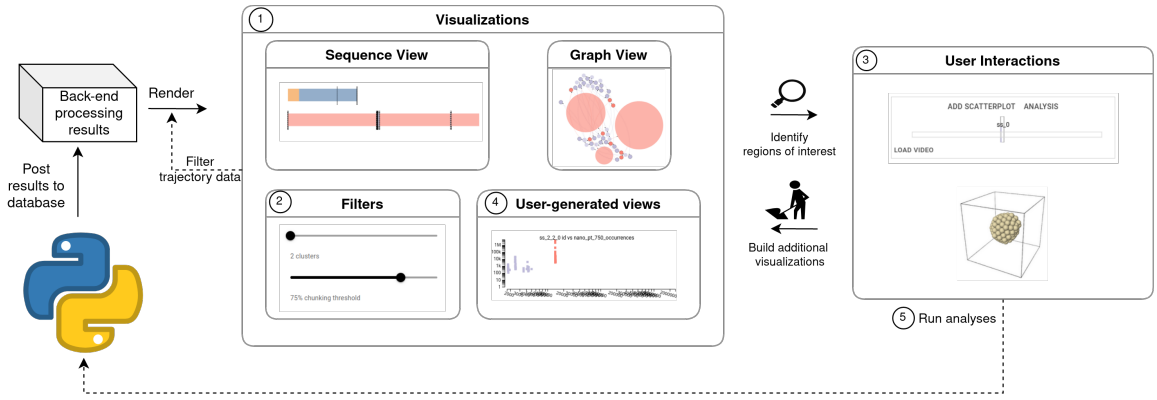

Figure 4.1: My Visual Analytics System Is Designed to Provide Experts with an Easy Way to Interact with the Trajectories They're Interested In. (1) the User Explores the Data-set, Using the Sequence View to Facilitate Selections, and the Graph View to Understand Relationships Between States; (2) the User Can Utilize Various Filters to Further Aid Their Analysis. (3) Once a Region of Interest Is Identified, the User Can Get a More Detailed Look in the Sub-sequence View. From the Sub-sequence View, They Can (4) Generate Additional Visualizations, and (5) Perform Various Analyses That Get Saved to the Database.

4.1 Design Objectives and Tasks

Each component of the pipeline went through an iterative design process with our domain expert who works in high-energy physics at a national laboratory; together, we identified a set of eight analytical tasks that I set out to simplify through the use of NeoMDWeb.

T1 Classify basins and transition regions in individual trajectories. Studying transition regions is critical to domain experts because the states within them can reveal how the structure of the atoms change between energy basins. Even with the help of PCCA clustering, which helps identify kinetically similar cluster within the trajectory, there are simply too many states for an expert to sift through manually - a computer aided automatic detection could save experts a lot of time.

T2 Identify commonly repeated states within one trajectory. Commonly repeated states can be used to further identify states worth exploring in detail. There is a need to be able to identify states that occur often within a transition region, throughout a PCCA cluster, and throughout the trajectory. Providing an interface that can seamlessly display the occurrence of a state within these levels of detail can save a lot of time for analysts, as without an intuitive visualization, determining when and how often a state occurs is very tedious, requiring experts to write convoluted and often disposable analytic scripts.

T3 Find states in common between trajectories. Through studying an ensemble of trajectories (e.g., simulations at different temperatures) and identifying a state common to all of them, a hypothesis can be formed about a particle's structural behavior in a general sense. Comparing entire trajectories manually

is often done in the temporal domain on a few variables - providing an intuitive way to compare trajectories visually in multiple dimensions could lead to new insights.

T4 Compare sub-sequences. Identifying structural similarities between sub-sequences of trajectories can also be used to form a hypothesis on a particle's behavior, especially when comparing novel data to controlled data. Moreover, providing a way to compare small regions of different trajectories can lessen the cognitive load on the analyst and lead to insights provided by details that are otherwise overlooked in a coarse-grained analysis.

T5 Run mathematical analyses on user-defined attributes. These analyses include calculating the Nudged Elastic Band (NEB) calculation, Common Neighbor Analysis (CNA), and many other structural computations. The analyses are stored in the Neo4j database and can be visually displayed within the framework in future analysis sessions. Providing an easy to use interface that can run multiple analyses asynchronously and save the results in a central location encourages collaboration between analysts, ultimately leading to more productive and meaningful analytical output.

T6 Automatically cluster and simplify raw trajectory data with Perron Cluster Cluster Analysis (PCCA). Manipulating trajectory data into the various formats that PCCA libraries (Deuflhard and Weber, 2005) use is not trivial, and providing an interface to do this can save time in future analyses. Prior to this work, no pipeline existed for extracting and visualizing simulation data generated by ParSplice.

T7 Avoid calculating unnecessary quantities throughout the data-set As

mentioned previously, most MD simulation data is repetitive and simply propels the simulation forward without providing any real insight to experts. Saving computational resources is of utmost interest to researchers, and is arguably the most important objective of my system; even with very generous computational resources, it is important to utilize each and every CPU hour provided to its utmost potential. Blinding running analyses on entire trajectories is likely to lead to noisy data, as well as waste computational resources.

T8 Provide a way to easily identify states that characterize a trajectory/cluster. Since states can be very similar yet have different IDs, it is important to provide an easy way to identify states that occur often throughout a cluster and a trajectory. Identifying the characteristic states of a MD trajectory helps shape an analyst's hypothesis about a particular reaction chain.

4.2 Architecture

The system is designed as an array of micro-services, allowing for greater flexibility between modules (Figure 4.2). Thanks to the micro-service architecture, DBExtract can be replaced with a program that can extract MD simulation data from other simulation software, placed in a format familiar to NeoMD, and be analyzed in the front-end.

NeoMDWeb's back-end is split into a RESTful API and a user-configurable amount of background workers provided by Celery; these background workers can be used to scale the application up to serve a large number of users, or as a means to run multiple expensive computations at the same time. Splitting up the back-end this way allows for a deployment scheme that can take full advantage of a High Performance Computing (HPC) cluster; the main back-end can be deployed on one node, and it

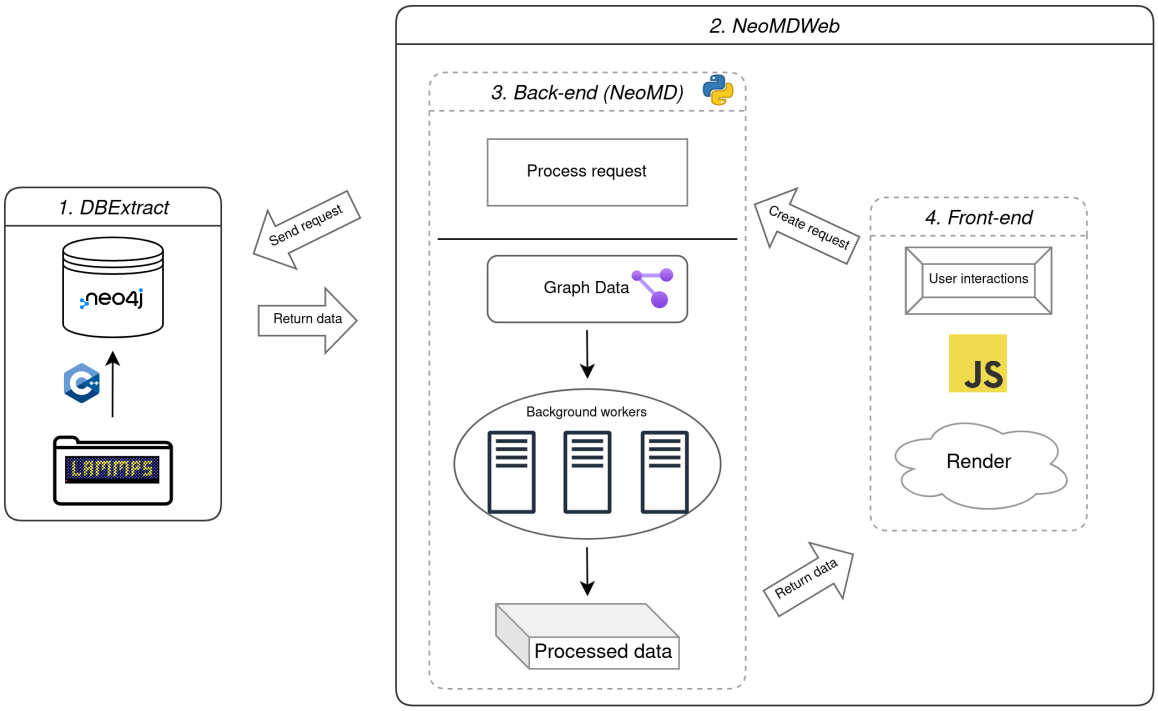

Figure 4.2: The Overall Architecture of the System Is Displayed. (1) a Molecular Dynamics Simulation Is Generated, and the Data Is Stored Somewhere on a HPC Cluster. Then DBExtract Is Executed, and the Simulation Data Is Populated into a Neo4j Database. (2) NeoMDWeb Can Be Launched, and the Data Is Available for Use - (3) NeoMD (the Back-end) Can Be Leveraged to Run Computations on the Data Using Background Workers Provided by Celery, and (4) NeoMDWeb (the Front-end) Is Used for Displaying the Results of the Computations, as Well as the Data-set Itself.

can communicate to other nodes hosting the background workers.

DBExtract and the Neo4j database are also designed to reside on a HPC cluster to meet the immense storage and memory requirements that long-duration simulations often require. NeoMDWeb's back-end should also reside on the same cluster in order to shorten back-end communications, to again meet the memory requirements that are common in long-duration trajectories. On the other hand, the front-end server can be launched as an application on the end-user's computer, using a SSH tunnel to connect to the cluster for better security.

4.3 DBExtract

DBExtract is a stand-alone C++ program originally included in ParSplice that I modified to populate Neo4j databases in parallel. When DBExtract is executed, it reads a simulation's trajectory file and extracts information from the corresponding simulation's LAMMPS data files. DBExtract first processes the trajectory in sequence before adding each state's corresponding atoms in parallel using OpenMPI.

DBExtract handles symmetrical transitions in an intelligent way in order to shorten the sequence and ensure the accuracy of the PCCA. DBExtract separates transitions from re-mappings of states with their canonical representations (explained in Chapter 2.4). To get the sequence itself, the back-end follows the relation named after the name of the simulation; to get symmetries, it can query whether or not a state has a "is canonical representation" relation. This relation is used during NEB calculations; as discussed previously, NEBs can only be performed between states that share the same atomic labels - this labeling lost during the remapping process, and must be kept separately.

Removing symmetrical transitions can make a huge difference in larger data-sets - besides removing redundant data, as well as simplifying and correcting the PCCA calculation, it can lead to large reductions in data-set sizes. In one of the test data-sets I used, with approximately 21,000 transitions and 2,600 unique states, removing symmetries cut down the number of transitions to approximately 16,000 and the number of unique states to approximately 2,000, yielding a near 25% reduction of the entire data-set. In a larger data-set, 8,000,000 transitions were cut to only 5,500,000, yielding a reduction of 32%!

The ultimate vision I have for DBExtract is to bring all of the old MD data formats into the 21st century by placing them into a graph database. It goes without

saying that trajectory data makes the most sense stored as a graph, since trajectories mostly comprised of relationships between a set of unique nodes. If all MD data became standardized, it allow for the data to be accessible at any time without the need for specialized software, ultimately increasing the speed at which the data can be accessed.

4.4 NeoMD

NeoMD most directly fulfills T5 - it provides a simple Python interface for performing diverse operations on trajectories, such as quickly building trajectory matrices, used in Perron Cluster Cluster Analysis (PCCA) libraries (Reuter *et al.*, 2018; Scherer *et al.*, 2015), rendering static OVITO (Stukowski, 2010) visualizations of molecular structures over time, building ParaView Ahrens *et al.* (2002) visualizations of the data-set, and more. Critically, NeoMD provides an interface for running both the Kilmogorov-Smirnov test (Dodge, 2008), as well as the Nudged Elastic Band calculation (Henkelman *et al.*, 2000) on parts of the trajectory, which was a highly sought-after feature. Under the hood, NeoMD also uses the Atomic Simulation Environment (ASE) (Larsen *et al.*, 2017)- Python package to manage atoms and maintain compatibility with a wide range of MD formats. Its main role in the pipeline is to pre-process MD simulation data for the front-end to render.

When a user decides to load a trajectory, NeoMD first queries the database to retrieve the trajectory's sequence and ignores any states that are topologically equivalent, leaving behind only one copy; i.e, the state's canonical representation. It then re-indexes the sequence and builds a corresponding transition matrix, and then runs a generalized version of the PCCA (Deuflhard and Weber, 2005) algorithm implemented in the PyGPCCA library (Reuter *et al.*, 2018).

4.4.1 Neo4j Queries

The queries provided by NeoMD are specifically optimized for the structure that DBExtract produces. Through cleverly eliminating bottlenecks present in unrefined Cypher queries, as well as exploiting the nature of the data-set and graph databases in general, NeoMD is able to quickly provide information on states throughout the data-set without any processing lag. Moreover, the query builder frees end-users from knowing anything about the structure of the data-base, as the queries generated by NeoMD are based on inferences about the data-base automatically made by the library through probing queries.

4.4.2 Data-set Simplification

In order to provide a smooth user experience and to lessen the cognitive load of analyzing the data-set, NeoMD provides a sequence simplification mechanism that reduces hundreds of thousands of states to several hundred groups called **chunks**. The amount of simplification performed by the algorithm is determined by the simplification threshold parameter, set by the user, that determines if a state is interesting to the user or not by comparing its PCCA cluster membership probability (**fuzzy membership**) to the threshold.

If a state's membership probability is above the threshold, it is considered "unimportant" and added to an "unimportant" cluster, and vice versa for "important" states. The difference between "important" and "unimportant" chunks is that "important" chunks are split recursively into N pieces, and are continually split until each sub-chunk is no larger than a chunk size threshold, which is also set by the user. Grouping states in this way removes states that are likely to be part of a basin, while leaving behind states that belong to transition regions. This simplification serves not

only to reduce the computational burden of the system, but also the cognitive load on the user, since the need to search through thousands of states for transition regions is lessened. Without the simplification algorithm, the system grinds to a halt, and yields extremely cluttered and difficult to read graphs. Ultimately, the simplification algorithm fulfills T7.

4.5 NeoMDWeb

4.5.1 Back-end

The back-end portion of the server software is written in Python to leverage NeoMD's capabilities and is powered by FastAPI (fas, 2019). A number of RESTful endpoints are provided for easy programmatic access to a variety of common MD analysis tasks - NeoMD provides the building blocks, while NeoMDWeb puts the pieces together. In the future, the front-end could easily be replaced without the need to re-write any of the computations, greatly improving the longevity of NeoMD and its ecosystem. An interface to Celery (cel, 2016) background workers is provided, allowing the back-end to be scaled indefinitely if placed into a production environment with larger data-sets and higher user demand. The Celery workers enable the asynchronous processing of background tasks while the user explores the data-set. Data properties are displayed as they are calculated, giving users immediate feedback on the quality and direction of their analysis.

4.5.2 User Interface: Overview

Once a trajectory has been processed by the back-end, a overview + detail (Cockburn *et al.*, 2009) interface is loaded (Figure 4.3). The sequence view serves as the overview; it provides the user with a horizontally-stacked bar-chart view of the tra-

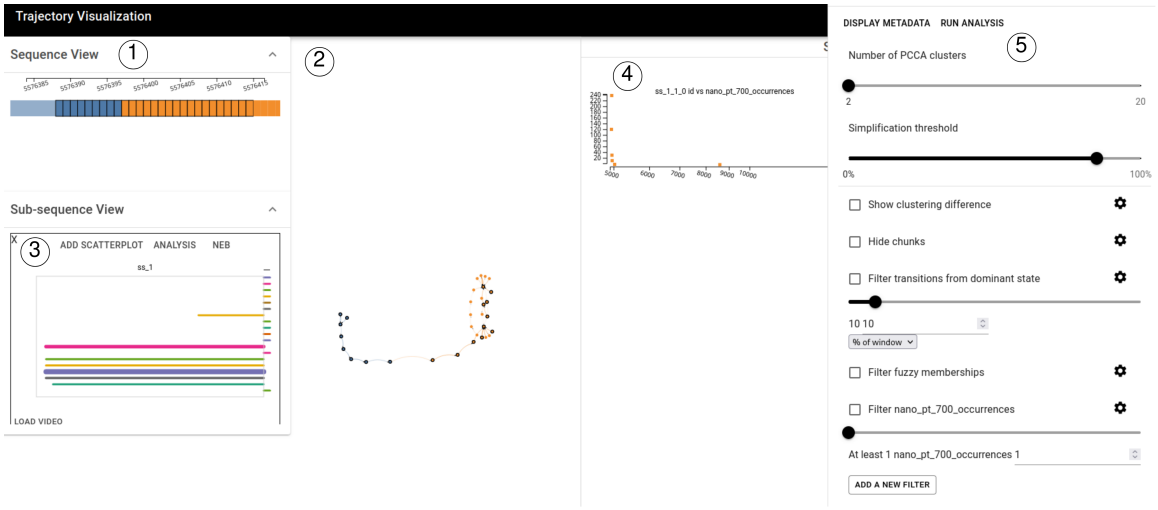

Figure 4.3: An Overview of the Entire User Interface Is Shown. (1) Is the Sequence View, Where Data Is Shown Temporally to Give Analysts an Idea of Where in the Trajectory the States Shown in the Graph View (2) Occur; The Graph View Focuses on the Relationships Between States; (3) Is the Sub-sequence View Which Shows the User's Previous Selections; (4) Is the Scatterplot View That Is Generated from the Users Selections, and (5) Is the Control Drawer Where the User Can Apply Filters to the Various Views in the System.

jectory, ordered by the time-step of each state. Upon zooming, the sequence view eventually turns into a pixel-based representation of the sequence, see Figure 4.5. The graph view is rendered in the main view-port; it provides a node-link diagram of every unique state within the ensemble of trajectories, positioned using the *d3-force* library provided by D3.js (D3, 2011).

Both views represent the simplified sequence generated by NeoMD, with distinct goals in mind - the sequence view highlights the contents of each time-step, while the graph view focuses on the relationships between each unique state within the ensemble of trajectories. Ultimately, these two views jointly fulfill the requirements outlined in the previous section - the graph facilitates T1, T3 and T4, while the overview facilitates T2, T3, and T5. The interactions available in both views fulfill T5, and both also display the same clustering, fulfilling T6.

The scatter-plot matrix view is based on the traditional way of analyzing tra-

jectories, providing a familiar X-Y plot with filtering and selection capabilities not available in simple PNG images generated by commonly used libraries like Matplotlib or GraphVis. The scatter-plot aims to define the trajectory through the properties of the data-set, rather than focusing on the relational/temporal dimensions of the trajectory; it serves as the detail view in NeoMDWeb.

The sub-sequence view is designed to help the user keep track of all of their selections made in any arbitrary view, and it opens up the possibility of running analyses on small portions of the data through the interactions available within the sub-sequence view's timeframe.

Finally, the NEB view is a simple line plot to demonstrate the results of a NEB calculation. It renders each state as a function of its time-step and energy level.

4.5.3 Data Representation

The colors assigned to nodes and parts of the sequence correspond to the clustering generated by NeoMD; these colors were provided by Colorbrewer's 12-class qualitative data-set (col, 2003). Groups of high cluster membership probability states are rendered proportionally to their size; the graph view represents them as large nodes, while the sequence view represents them as bars. Clusters considered "important" are rendered with a dashed outline and 100% opacity, while "unimportant" chunks are rendered with a lower opacity with no outline. States gain a solid outline when hovered, and the cursor changes to the classic selection style when they can be clicked.

4.5.4 Sequence View

The sequence view (Figure 4.3(1)) displays states and chunks in a linear fashion, allowing users to easily select parts of the trajectory for analysis by using common modifier keys (SHIFT and CTRL), fulfilling T5. Hovering the mouse over a state

highlights all of the occurrences of that state within all trajectories, fulfilling T2; hovering also automatically pans the camera in the graph view to the corresponding node within the trajectory. The scale is set to range from 1 time-step to the entire length of the longest trajectory loaded, setting states to be the smallest width possible, and scaling the chunks up based on the size of each individual node. This scale conveys the immense size of some chunks, but has the effect of sometimes rendering individual states very small when zoomed out, but this issue is mitigated by the semantic zooming feature.

The sequence view does not readily show rare transitions, transitions between basins, or states that are within multiple trajectories. To overcome these limitations, I implemented the graph view which utilizes a novel layout mechanism based on the cluster of each state / chunk.

4.5.5 Graph View

The graph view (Figure 4.3(2)) is intended to qualify the user's understanding of the relationships between states; a node-link diagram comes as a natural metaphor to express these relations. There are a variety of options available to the user through a context menu, accessed through right-clicking the graph - they toggle various features of the graph such as rendering arrows on links, highlighting the currently selected node by hiding the rest, and the "separate trajectory" option. Just as the sequence view, the scale of the graph's nodes are set to be based on the size of the timesteps contained in the chunk, ranging from the default 5 pixel radius for individual nodes to a 125 pixel radius for the largest possible chunks.

The dashed arrows in the graph view represent the temporal adjacency of chunk / states that are not directly next to each other in the sequence - if a node has more than a single time-step difference between its neighbor, their relationship is rendered with

a dashed stroke. This maintains temporal cohesiveness with nodes, and highlights the nodes that are actually temporally adjacent.

Separate vs Conjoined Trajectory View Modes

When "separate trajectory" is toggled on, a general position for each trajectory is calculated using a layout inspired by the group-in-the-box algorithm first conceived in Rodrigues *et al.* (2011), and nodes that are shared between trajectories are duplicated for each trajectory that they are a member of. Once the trajectory has been localized to a part of the visual space, each cluster within the trajectory is assigned a local x position in which its nodes will attempt to remain. This leads to a graph where nodes that have transitions between clusters are rendered in between each cluster column, thus fulfilling T1, i.e these nodes are within a trajectory region.

On the other hand, if the "separate trajectory" rendering mode is turned off, each trajectory is assigned a local x position instead, acting as the cluster did in the other rendering mode. This creates a graph that highlights which nodes are shared between trajectories - these states are rendered black, and the transitions that point to them demonstrate the clustering in which they participated in depending on the trajectory that is being considered; ultimately, this functionality satisfies T4. Groups of states detected by the simplification procedure are the only elements within the graph that have a fixed layout - they are organized in the y direction in the order in which they appeared throughout the trajectory. While both views are useful alone, their mutual interactions make them powerful.

4.5.6 Scatter-plot

The scatter-plot matrix (Figure 4.4) fulfills T4 - it enables the user to compare the attributes of various sub-sequences as they evolve over time, or how groups of

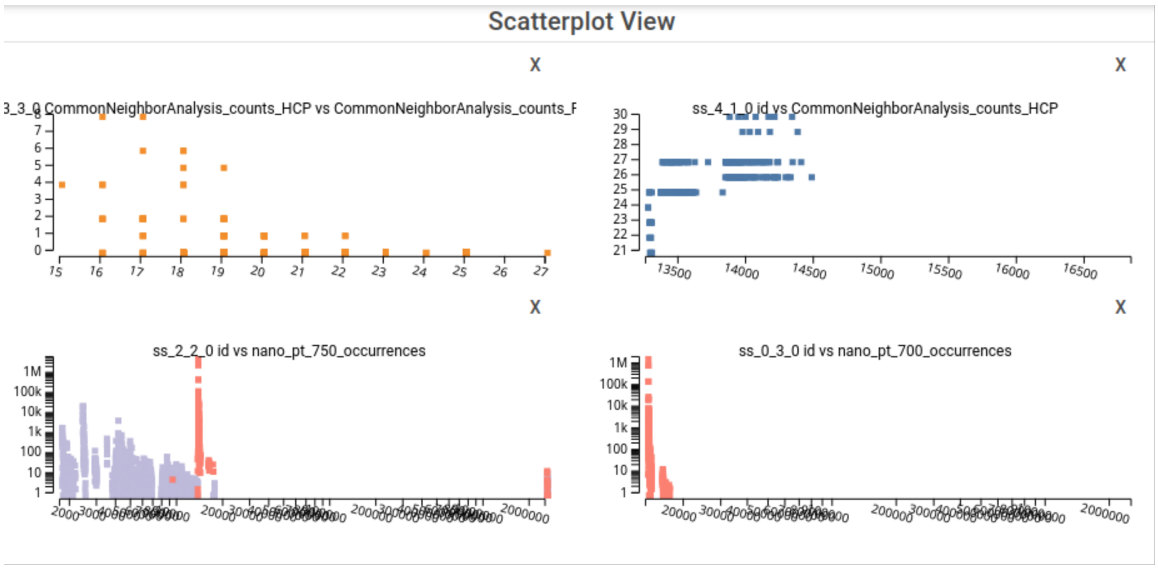

Figure 4.4: The Scatter-plot Matrix Featuring Scatter-plots Derived from Two Different Trajectories. The Top Two Focus on Structural Properties of Atoms, While the Bottom Two Look at the Occurrences of Various States Throughout the Two Trajectories Selected. The Bottom-right Scatter-plot Reveals That the Pink Cluster from Nano-pt-750 Shares States with Nano-pt-700. This Demonstrates the Power of the Scatter-plot Matrix in Making Inter-trajectory Comparisons.

states differ from each other. It also has the power to compare entire clusters / trajectories - since the representation of states in this view mode is relatively simple, the scatter-plot itself can render a large number of states. The fact that properties are decoupled from their trajectory also allows the in-depth comparison of trajectories, as demonstrated in Figure 4.4. Thus, the scatter-plot matrix also fulfills T3.

The scatter-plot matrix is built entirely by the user - they can either choose to render the entire data-set through clicking the + button at the top of the window, or through the ADD SCATTERPLOT button in the sub-sequence view(s). The X and Y values for each scatter-plot can be changed by right clicking the plot and selecting the desired attributes within a context menu.

4.5.7 Coordinated Interactions

Hovering above a state in any view triggers the hover action in all of the other views, i.e, the sequence and scatter-plot views highlight all of the occurrences of the given state while the graph view moves the camera to the state. This coordinated mode of interaction works between scatter-plots as well, making it exceptionally easy to see states in common between selections, fulfilling T4.

Pressing and holding SHIFT while clicking and dragging the mouse allows the user to select any arbitrary subset of the trajectory with a brush, and selections are continuously made until the user lets go of the SHIFT key (Figure 4.6). Pressing and holding the CONTROL key while clicking on states selects them individually, allowing for fine grained selections between states that are not necessarily spatially adjacent in any view. These behaviors work similarly in any view, allowing the analyst to select data based on its occurrence in the trajectory, as well as selecting states based on their relations with each other.

Clicking on states without holding any modifier key in any view opens a modal that displays more information, including the attributes of the state and its structural rendering. Pressing the left or right arrow keys after hovering on a state in any view moves the graph view to the next state in the temporal sequence, as well as highlighting it in the sequence and scatter-plot views.

Semantic Zooming

The most important coordinated interaction is the sequence view's ability to provide semantic zooming for the entire system, shown in Figure 4.5. When exploring the data-set, zooming into any important cluster breaks it down into smaller chunks, and zooming in sufficiently enough ultimately leads to individual states being shown.

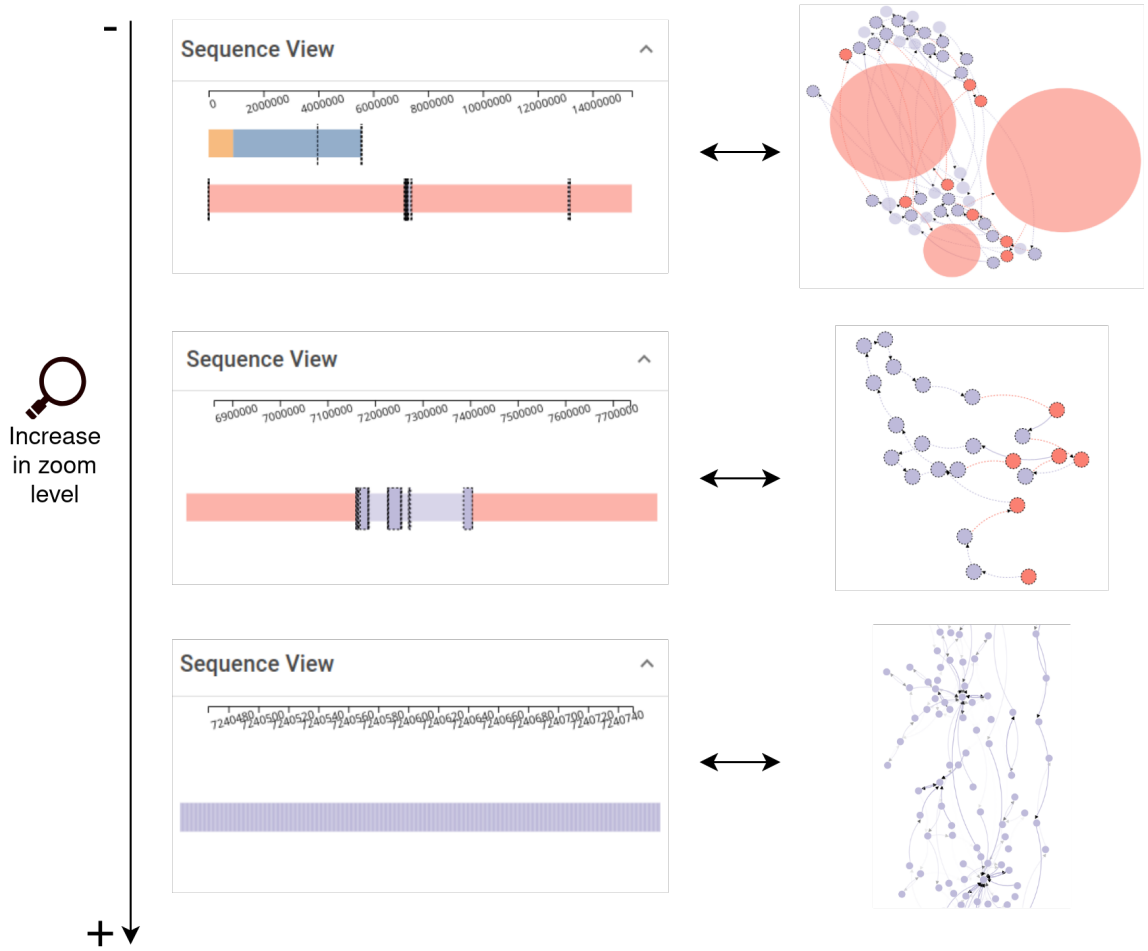

Figure 4.5: Example of the Semantic Zooming Feature Being Used to Focus on a Transition Region Detected by the Simplification Algorithm. As the User Zooms in, the Region Expands to Show Multiple Chunks That Are Broken down Further until Individual Nodes Are Reached.

Zooming out far enough merges clusters together, and removes individual states from the view in order to ensure a smooth user experience. This is mimicked in the graph view - the graph view shows only the important chunks/individual states that are within the sequence view's visible range, allowing for fluid user interactions, as well as simplifying the process of finding and selecting interesting portions of the trajectory.

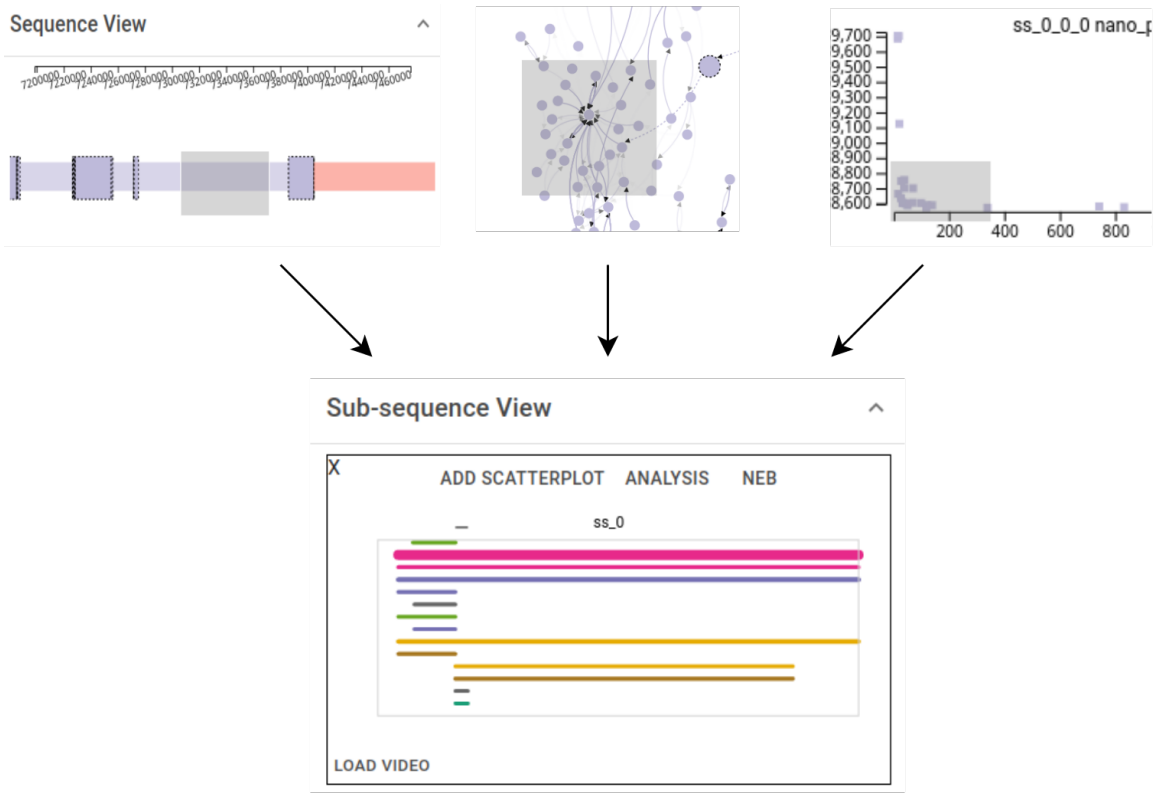

Figure 4.6: Some of the Brush Interactions That Create Sub-sequence Views Are Displayed. On the Left, the Sequence View Is Brushed to Create a Sub-sequence View, in the Center, the Graph View Is Brushed, and on the Right, a Scatter-plot Is Brushed. All of These Sub-sequence Views Contain the Same Functionality When It Comes to Analysis; However, NEBs Can Only Be Performed on Selections Made in the Sequence View Because Time-step Information Is Preserved When Making a Selection in the Sequence View.

4.5.8 Sub-sequence View

Selections made with the modifier keys in any visualization create a sub-sequence selection in the sub-sequence view; see Figure 4.6. This view shows the relative temporal occurrences of the states that were selected, and presents the user with several buttons that open modals for further analysis. Each line associates with one unique state; the width of one line shows the relative occurrence period (from the earliest to the latest timesteps) and the thickness of one line represents the occurrence of a state within the given timeframe.

When selections are made within the sequence view, they are represented by lines that span the entirety of when the states within the selection occur within the trajectory. The different colors help identify unique states within the sub-sequence selection, and they are also coordinated as with any other view - hovering/clicking states in the sub-sequence view finds its equivalent in the other views. This view helps users search for similar selections within the currently loaded trajectories. This also helps reinforce the difference between the sequence view and the other views - the sequence view is meant for exploring the data temporally, while the other views focus on different dimensions of the data.

The ANALYSIS button opens a modal with options to run the Kilmogorov-Smirnov test on the selection against a distribution, a cosine similarity calculation if multiple selections are made, as well as the option to run OVITO analyses, custom Python code and Neo4j queries on the selection. The results of every analysis can be rendered on the screen by checking the corresponding "save results" checkbox in each modal window; these results are displayed underneath each sub-sequence view. The ADD SCATTERPLOT button creates a new scatter-plot from the group of states selected - multiple scatter-plots can be made from one selection, allowing the user to build a scatter-plot matrix for any attribute loaded in the trajectory.

Finally, the NEB button allows the user to run the NEB calculation on selections that were made in the sequence view (i.e selections that preserve temporal order). The results of each calculation are added to the NEB view, which is simply a list of line charts containing the NEB data - these charts describe the minimum energy paths along the course of the selection; see Figure 4.7. The NEB implementation used in my system is provided by LAMMPS.

Each sub-sequence view also contains an OVITO visualization provided by the back-end, allowing the user to see the physical structure of the atoms changing over

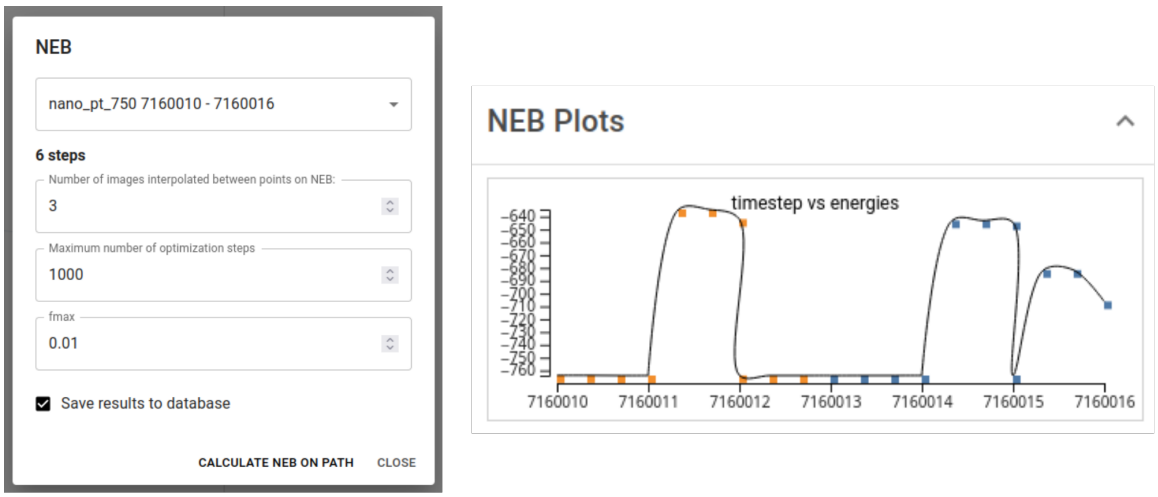

Figure 4.7: The NEB's Parameters Can Be Tuned Using the Modal That Pops up upon Clicking a Sub-sequence View's NEB Button (This Is Only Accessible If the Selection Was Made Within the Sequence View). The Calculation Is Performed by a Background Worker, so the User Is Free to Continue Exploring the Data-set While the Computationally Complex NEB Operation Completes.

the subset(s) selected. Since the rendering of each frame is computationally expensive, if the length of the sequence exceeds an arbitrary number (100 states), a LOAD VIDEO button is shown instead so that the server is free from wasting time computing useless data.

In order to contextualize the user's visible range in the sequence view, a box corresponding to the extents of the sequence view's view window is overlaid on each sub-sequence view. This context-box functions as a focus + context (Cockburn *et al.*, 2009) view and provides a way for the user to understand their viewpoint in relation to the selections they have made.

All of the calculations provided by the sub-sequence view are implemented asynchronously - Celery handles the task in the background while the user is free to interact with the rest of the system, allowing for a fluid user experience.

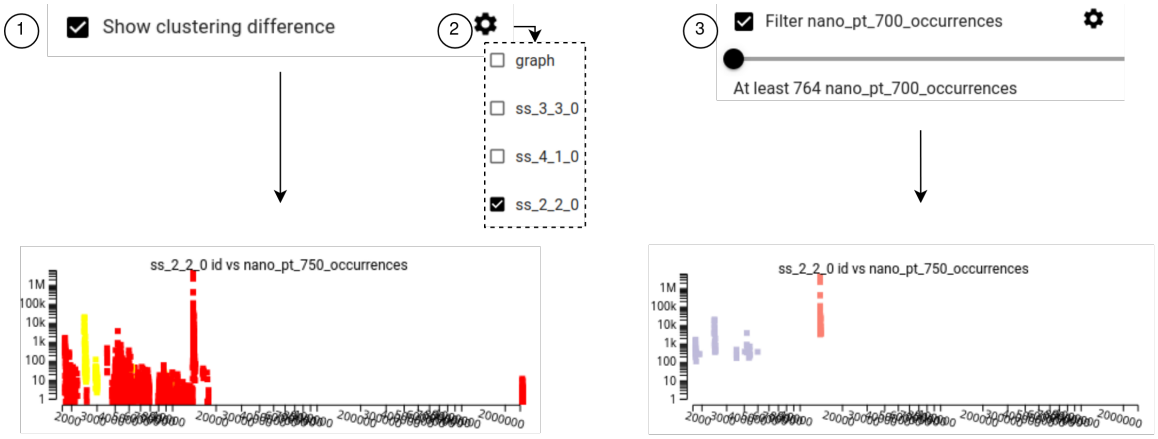

Figure 4.8: An Example of How the Filtering System Works with a Scatter-plot. On the Left, (1), the Clustering Difference Algorithm (Zhang *et al.*, 2016) Is Applied to the Scatter-plot to Describe Which Clusterings Are Stable. (2) Demonstrates the Selection Menu in Which the User Can Choose What Visualizations the Filter(s) Are Applied To. On the Right, (3), a Simple Opacity Filter of Nano-pt-700 Occurrences Is Applied, Removing All States That Occur Less than 764 Times. Not Pictured Is the Ability to Chain Filters Together to Get a Finer-grained Look at the Data.

4.5.9 Control Drawer/Filtering

When the user clicks the menu button at the top right of the screen, a drawer slides open and presents an accordion element for each trajectory as well as a properties accordion. The property accordion contains a table with a list of toggle-able properties; toggling a property loads/unloads the selected property for each state in each trajectory currently loaded. If a property does not exist for a given state, they are simply not rendered if they are chosen in the scatter-plot view - the undefined property is also not displayed within any tool-tips if hovered.

Each trajectory accordion contains options to change the simplification level of the trajectory, the ability to change the number of clusters shown, and a variety of filters. Every filter can be applied to each trajectory and view individually, allowing for powerful, fine-grained interactions. Moreover, the gear next to the filter's name allows the user to apply the filter on any arbitrary plot, allowing for powerful comparisons to be made in the scatter-plot matrix - the user can take the same sub-sequence, create

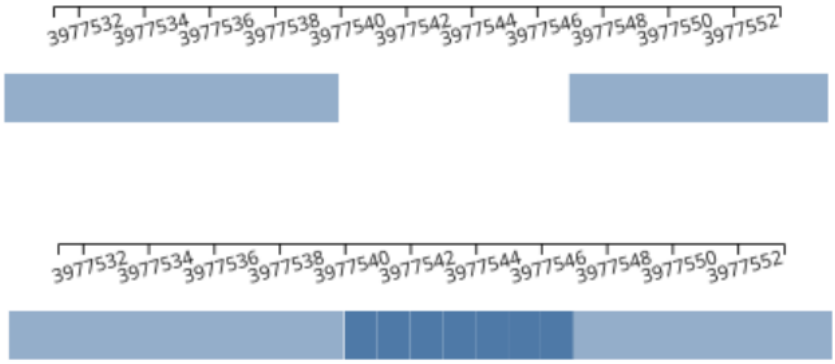

Figure 4.9: An Example of the Dominant State Filter. The Top Sequence View Is with the Filter Applied, and the Bottom Is with the Filter Switched Off. This Small Region of Individual States Does Not Have Any Transitions to/From the Dominant State, so They Are Rendered with Zero Opacity.

two scatter-plots, and apply different filters to gain a greater understanding of the nature of the data.

Several filters are loaded automatically, and the user is free to create their own based on the attributes of each trajectory; these filters map a state's opacity based on the user's criteria. The attribute filters are relatively simple - they allow the user to determine a cut-off point and only render states that are above/below or within the range set by the user. Notably, I implemented a filter that attempts to further define transition regions using the clustering difference algorithm described in Zhang *et al.* (2016); "unstable" regions, or regions where states tend to change their cluster membership depending on the number of clusters, are colored red, while "stable" regions are colored green, yellow dots are states in between (Figure 4.8).

The dominant state filter iterates through the sequence with a user-defined window, and tries to determine how many times the states within the window transitioned to/from the dominant state of the cluster (i.e, the state the occurs most frequently within a cluster). This filter is designed to help co-locate states that accompany the dominant state of a cluster (Figure 4.9).

Interactions are not suspended while the graph view is rendering - the user is free to operate the system and perform various analysis tasks using the sequence view. For instance, the user can select one or more parts of the trajectory within the sequence view and directly compare them, as well as run all the analyses provided by NeoMD on any arbitrary subset of the trajectory.

Chapter 5

PLATINUM NANO-PARTICLES: A CASE STUDY

Prior to beginning the analysis, all of the data was pre-processed due to the immense computational workload it entailed. All three data-sets were loaded using DBExtract the previous day, and PCCA was applied to each data-set, using cluster sizes from 2 to 20; extracting the data using DBExtract took about an hour on average, and the PCCA calculation ranged from 1 to 2 hours per trajectory. The results were cached in JavaScript Object Notation (JSON) files to be used by the analyst, as PyGPCCA was not used with PETSc and SLEPc. The aforementioned libraries were not used due to linking issues, which unfortunately meant that I could only leverage the slow Python implementation. Once the PCCA was completed, the optimal cluster count was reported by PyGPCCA, and was used in each trajectory analysis.

Since the data is loaded completely in memory, high-performance computing (HPC) clusters were used to be able to manipulate the large data-sets. Due to the difficulty of managing dependencies on different machines and the lack of secure container technologies available for HPC clusters, *charliecloud* (Priedhorsky and Randles, 2017) was used to containerize the workflow.

The data itself is comprised of long-duration versions of the data-sets first described in Huang *et al.* (2017). Even with the use of HPC clusters, there are still data-sets that I have not yet analyzed due to their size. The nano-particle simulations at higher temperatures are 4 times larger in the case of the 800K simulation, and nearly 10 times larger in the 900K simulation, taking up 14 and 32 GB respectively. In comparison, the data-sets that were analyzed, 700K and 750K, take up 1GB and 3.5GB respectively. Despite the fact that these data-sets could not be loaded,

my system is still the first to allow the analysis of long duration molecular dynamics simulation, as these data-sets did not exist until recently, and are confined to the domain of materials science.

The ultimate goal of each analysis was to identify groups of states that made up transition regions and to characterize/summarize their structural properties to get insight into the mechanism under which the molecule changes its structure. The domain expert I worked with wanted to get a feel for the system itself and explore a relatively small data-set on its own, so he opted to load the 700K platinum nanoparticle data-set first.

5.1 Nano-pt 700K

To begin the analysis, he sets the PCCA clustering algorithm to cluster the data-set into 2 to 20 clusters, expecting an optimal clustering amount to become apparent through analysis. He immediately notes, "It is impressive that the system remains so responsive despite the size of the data-set." He then sets the clustering amount from 2 to 4, 6, 10, and then to 20, seeing that the trajectory stays in the same energy basin for most of the trajectory, starting from approximately time-step 1000000. He then sets the clustering amount back to 4 clusters, stating that "this clustering amount seems to be the best balance between being detailed and noisy; there are a few micro-regions within the transition from the left colorful side to the right teal side I'd like to explore." He also sets the simplification threshold to 60%, in order to cut back the amount of useless states.

The next thing he does is make large selections over the various clusters; despite having 4 clusters selected, it seems like there are two primary clusters that characterize the data-set. He presses the SHIFT key and brushes over the two major clusters and builds a scatter-plot for each cluster. He sets the X attribute to be the ID of the state,

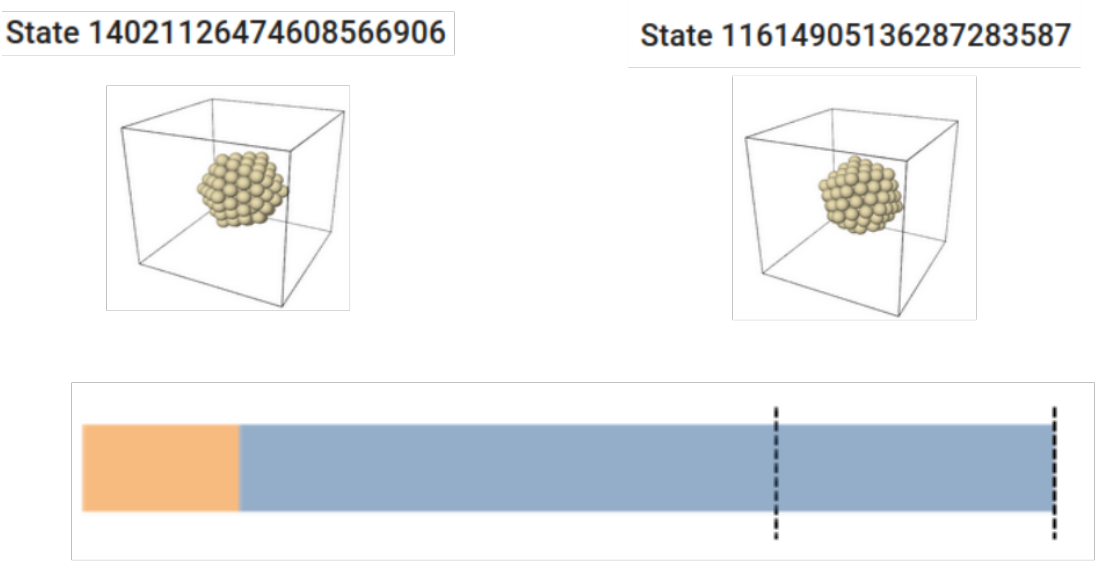

Figure 5.1: The Characteristic States for Nano-pt-700k. The Blue Characteristic State Seems to Be a HCP Structured Particle Based on Its Relatively High HCP Count, While the Orange Characteristic State Seems to Be a BCP Structured Particle.

and the Y attribute to be the number of occurrences in the trajectory. Once again, he notes the speed and fluidity of the system despite looking at over 8 million transitions. Once the two scatter-plots are built, he applies filters that cut off states that occur more than 500 times - he states, "I am looking for states that characterize the clusters, as well as states that are rare - they might be interesting for further analysis." He notes that the very first cluster that appears does not really contain any interesting states that characterize the cluster - this makes sense as the blue cluster only occurs in the first several thousand transitions of the trajectory. He individually clicks a few states to look at their structures - he notes that the states that occur the most in each cluster are all very similar, dictating the need to drill down into the data further, as well as simplify the number of clusters. Thus, he switches the number of clusters back to 2, which was the original suggestion given by PyGPCCA. He sets the simplification threshold to 85% to account for the fact that there are only two clusters splitting the membership probabilities instead of 4.

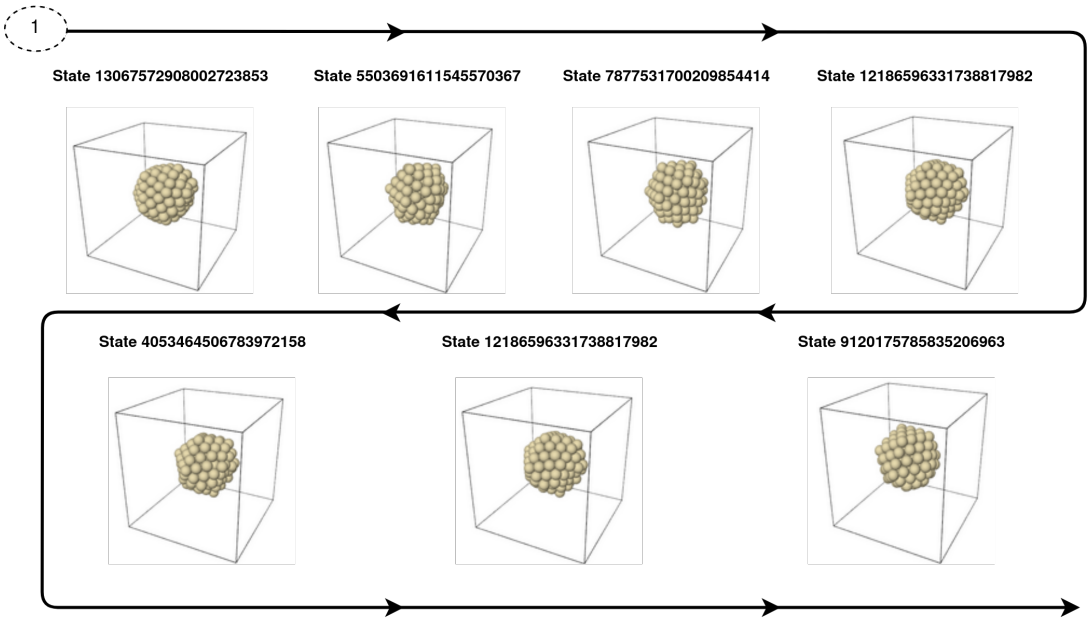

Figure 5.2: The Transition Pathway Discovered in Nano-pt-700. Note the Structural Change Through the Series of States.

He then selects all of the states in the scatter-plot with more than 500 occurrences, and runs the Ackland-Jones (Ackland and Jones, 2006) and Common Neighbor analyses to gain more insight on the data. He repeats this for both scatter-plots, the one that characterizes the left hand side of the trajectory, and the one that characterizes the right. He then starts building filters on the scatter-plots based on the structural properties of the selections; he sets up a filter for the max filter on the HCP counts calculated by both analyses, as well as a filter on the OTHER count. After applying the filters gradually and noting the states that both occurred numerous times, and had a structure different from the overall structure of the trajectory, he identifies the states that characterize each cluster. As shown in Figure 5.1, State 11614905136287283587 characterizes the blue cluster (as well as most of the run), and state 14921126474608566906 characterizes the orange cluster to the left. There are a number of states that are similar to each characteristic state, but they are omitted here.

He then starts to zoom into the first transition, and comments "the graph view changing as I zoom in helps a lot to keep my analysis focused." He zooms into the region between timesteps 899650 to 899700, where the orange clustering begins to merge into the blue clustering. Once he zooms in far enough, individual states are visible for the first time - he immediately begins clicking the states that seem to have large number of neighbors in the graph view; these are therefore the most recurring states within the region selected (T2). He also notes, "the graph view very intuitively shows the most frequently occurring states within this region; I was immediately drawn to this state that was central to all of the other states." The characteristic states for each cluster have been already discovered by the scatter-plot, but the transition region analysis shows the exact mechanism in which the transition occurs. The analyst decides to select some part of the sequence that starts before the indicated important cluster to slightly after to see how the trajectory changes. He runs the Ackland-Jones and Common Neighbor analyses on this sub-sequence. He also elects to load the OVITO video to observe how the clustering changes over time, and it reveals the the transition region detected by the simplification algorithm does indeed correspond to a change in the particle's structure - the simulation starts in a relatively rare state, and progresses through a series of easy to-see transformations until it arrives at a state similar to the characteristic state of the orange cluster; this transformation is captured in Figure 5.2.

Finally, the analyst decides to run a NEB calculation on the transition path to confirm the results attained by visually inspecting the states. Indeed, as soon as the blue cluster begins, a higher energy barrier is observed - meanwhile, the states in the transition regions have low energy barriers in between, with saddle points corresponding to the energy needed for a transition.

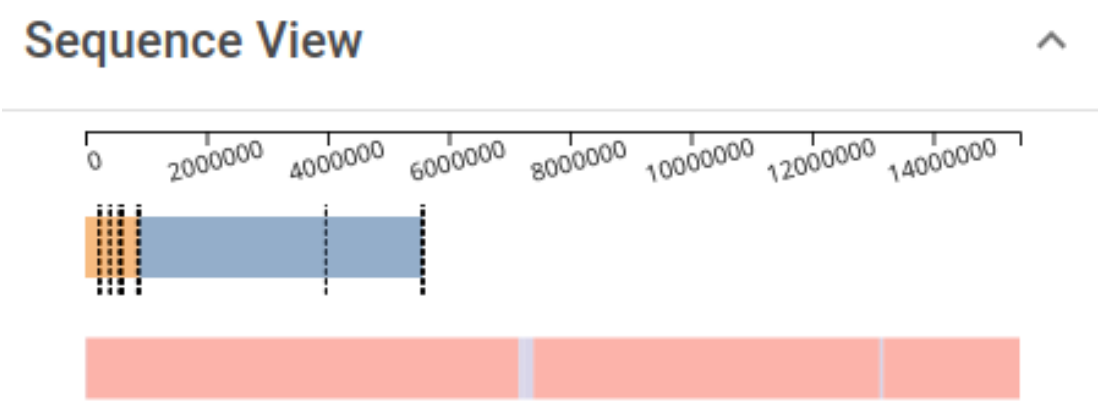

Figure 5.3: A Look at the Clustering for Nano-pt-700k, Shown as the Second Bar with Pink and Purple Clusters. The Trajectory Spends Most of Its Time Within the Pink Cluster, and Visits the Purple Cluster Momentarily Before Returning to the Pink, Visiting It Again Once More Before Continuing the Rest of the Run in the Pink Cluster.

5.2 Nano-pt 700K and 750K

Due to the relatively small temperature difference between the 700K and 750K simulations, the domain expert decided to explore the two simulations at the same time to compare them. Since he used the same data-set as before, the characteristic states for nano-pt-700 were known ahead of time - refer back to Figure 5.1.

In order to determine the characteristic states for nano-pt-750K, the analyst uses the same technique as he did for the previous trajectory. He creates two scatter-plots that cover the two major clusters in the nano-pt-750K trajectory - pink, which spans most of the trajectory, and purple that splits the pink into two separate sections; see Figure 5.3. He looks at the scatter-plots that show the number of occurrences for each state in both trajectories - unsurprisingly, the most common state in the 700K run is also the most common state in the 750K run. Noticing that a lot of the 750K trajectory has overlapping states with the 700K trajectory, he brings back the scatter-plots that cover the previous two clusters from the 700K run. After hovering over a

few states in the scatter-plot, it becomes obvious that the pink cluster in the 750K run corresponds to the blue cluster in the 700K run, and the purple cluster in the 750K run corresponds to the orange cluster in the 700K run. Thus, the characteristic states of the 700K run were the same in the 750K run; this was expected considering the fact that there was a minor temperature difference between the two trajectories.

However, the transition between the purple and pink cluster remains unexplained - the analyst decides to continue exploring the two trajectories. There are three transition regions within this trajectory, and it would prove for an interesting analysis to compare the three regions with each other, as well as the transition region discovered in the case study for the 700K run. He zooms into the region that splits the pink cluster into two chunks, and makes a selection that covers the left transition from pink to purple, and the right transition from purple to pink again, and first runs the CNA and Ackland-Jones analyses. He then sets up several simple cosine similarity computations between the two selections based on their structural properties. Their similarity score is returned, but it is a mere 10% - not enough to be a significant similarity - this ends up to hold true for the other pairs of transition regions.

The analyst then sets the simplification threshold to 100%, meaning the simplification is turned off. He zooms into the first transition region, and looks at the graph view to discover that indeed there is a transition from the characteristic state of the first cluster to the characteristic state of the second cluster, within approximately 40 timesteps. He comments, "I wish the transition region encompassed more of the states within the pink clustering to really show me the complete transformation between the two clusters." He identifies the characteristic state of the second cluster by simply hovering over and clicking a node in the graph view that is surrounded by neighbors, and he instantly recognizes the state as the characteristic state of the second cluster by its number of occurrences.

Then, the analyst pans over to the region where the pink cluster is split by the purple cluster, and finds the simplification algorithm has left 3 extremely rare states from the pink cluster as a part of the transition region. After these three states occur, the characteristic state of the purple cluster occurs shortly after. He repeats this process for each transition region, identifying the pathway in which the transitions occur, and finds that they all differ from one another, but he still wants to explore the relationship between the 700K run and the 750K run.

The analyst selects the first transition in the 700K trajectory, and makes an 750K occurrence count to ID scatter-plot, and zooms into the first transition region within the 750K sequence. As he hovers over the scatter-plot, and as the graph view jumps from node to node, he realizes that while the nature of the 750K transition is different from the first transition in 700K, both trajectories end up in the same set of states afterward. This is made apparent through the coordination of the scatter-plot with the graph view - the analyst is able to compare the properties of the 700K transition region with the relationships in the 750K transition region.

Since there are not many similarities between the two trajectories, there is not much left to do - the analyst classifies the rest of the transition regions, and finishes his analysis.

Chapter 6

DISCUSSION

The nano-pt-700K data-set demonstrated that NeoMD can be used to determine the characterstic states of clusters, as well as quickly determine transition regions - the analyst did not hesitate to zoom in on the region mentioned above to find the transition between the orange and blue clusters. Despite the fact that the analyst selected the region that extended beyond the area suggested to him by the system, the states suggested were exactly the states that were ultimately determined to be the transition states between the two clusters, thus qualifying the usefulness of the system in finding transition regions. The system allowed the analyst to visually and analytically verify the transition regions, and observe exactly how the particle changes structure during the transition in real-time. The fact that the clusters robustly characterized the meta-stable sets present in the trajectory aligns with the findings of Huang *et al.* (2017), verifying the usefulness of applying PCCA to nano-particle simulations.

The system also was effective in comparing multiple trajectories; the analyst was able to quickly find states in common between two trajectories, and make judgments on how transition regions differed between the two trajectories. The path similarity calculation saved the analyst a lot of time in making comparisons between selections - within seconds, a similarity score based on structural information was calculated between groups of states. Usually, these kinds of analyses are run through simple scripts, and are not reusable between simulations without altering a significant portion of their code; this is not to mention the immense cognitive load of keeping the particle structures in mind.

Despite the ease of analysis, there are a few weak points of the system that hin-

dered making completely fluid judgments. The primary weakness in the system was the structural view - it was very difficult for the analyst to compare states with one another and get an overall idea of the atom, since the structural view is a static rendering of one angle of the atom. Besides the structural view, there could be improvements with the general workflow within the system, as the analyst often noted that the interactions were clunky and still did not completely reduce the cognitive load of analysis. A better way to classify transition regions is needed, as the regions detected do not entirely capture all states within a transition region.

Chapter 7

CONCLUSION AND FUTURE WORK

7.1 Future Work

The case study demonstrated while the system drastically simplifies trajectory analysis, there is a still a large body of work that can be done to improve the system. The two major areas of improvement lie in improving the user interface, namely the structural view, and improving the back-end analysis techniques. The structural view's interactions were found to be lacking in depth compared to the interactions provided by OVITO's own native interface. A larger set of analysis options would also greatly enhance the user experience, and streamlining the process will be the among the first goals to reach in the future.

7.1.1 Visualizations / User Experience

Using an approach similar to the one proposed in Kincaid (2010) to highlight and stretch transition regions can improve the user experience when looking for regions of interest. However, this could be difficult to implement without modifying how the semantic zoom feature works in the sequence view; more work needs to be done before implementing this feature. Additionally, encoding the length of time the trajectory spends in a discrete timestep could aid in visually detecting interesting states; this was not implemented due to time constraints and the fact that extent selection relies on each state being a set length.

The analyst wished that he could select states through a searchable text-box where one could put in a desired value and have only matching states be displayed. This

could certainly save time in locating specific states, as filtering, panning and zooming can be too coarse to specifically locate states.

Another comment I received during the case study was the fact that the OVITO visualizations could not be modified at all - the user is only shown one angle of the atoms without any modifiers applied. In typical analyses, OVITO visualizations are rendered with a number of modifiers applied that encode attributes about the molecule's structure; not rendering these color encodings severely harms the ease of using the system. Adding this feature may prove to be difficult, as I am not sure how to provide an interface to OVITO through JavaScript without essentially writing my own, which is a huge body of work. Perhaps the best solution would be to just statically render the atoms, and provide a simple interface, similar to the analysis view to add modifiers. In order to get a good view of the entire atom, I change the rendering to slowly pan around the molecule before switching to the next in the sequence. Adding an easy-to-use interface to compare molecular structures was another highly-requested feature by the analyst - the current view is simply too small and far from other structural views for the user to really appreciate the differences between two structures, and clicking through modals to get information does not suffice.

The analyst also mentioned that the process of building a scatter-plot matrix manually was very tedious - adding an automated way to choose the most relevant properties of the trajectory could enhance the user's experience and save clicks. A brute-force approach that comes to mind is to simply build a Cartesian product of all of the properties currently loaded in the system and display the resulting matrix to the user. More intelligently, some properties such as the bounding box dimensions of the state and atom counts, which are unlikely to change throughout the course of a simulation, can simply be hidden and then the brute-force matrix method applied.

The graph view can be improved through clustering states in the graph by their attributes, which could lead to interesting insights. More time is needed to determine a graph layout that could be as information-dense as possible. Additionally, the graph can visually benefit from graph-bundling, where overlapping links are bundled together to reduce clutter, as well as circle packing chunks with their children.

Chunk interactions could be expanded in order to demonstrate similarities between chunks and their children. One method could be to highlight all similar chunks on hover, based on the state that occurs most often within the chunk. There should be a way for the user to select a chunk and automatically select all of its contents for analysis in the sub-sequence view.

The sub-sequence view could also be made more powerful - there could be an option to view the sub-sequence selection as a mini-graph view, in order to view the relations within easily. Moreover, the sub-sequence view could add interactions for the user to be able to get a detailed look at the states within without having to pan to the selection in the sequence view. This is to get around the rigidness of the simplification algorithm, as the analyst was often interested in viewing areas that were classified as non-important by the system.

A task-queue view for viewing background analysis tasks could inform users about the state of their queries; this feature, coupled with a smart cache system could significantly enhance the power and utility of NeoMD. The cache system could save the results of analysis for other analysts to view on their machines, saving computational costs. This cache system needs to be implemented for NEBs as well, since they are not available for the user to browse without directly interfacing with the Neo4j database. Currently, the only way to see an NEBs results is to run the same calculation again, which can take a considerable amount of time depending on the fidelity of the analysis requested. Additionally, more work needs to be done parallelizing and optimizing

certain computations in the back-end to provide a more fluid user experience, as the PCCA algorithm tends to take over an hour for modest-sized data-sets.

NeoMDWeb itself can be made to stream its results, but this could be difficult due to the nature of the computations. Because the PCCA calculation requires the entirety of the sequence to be loaded for it to complete, it is unlikely that the PCCA results could be streamed without the entire trajectory. However, the design of the system could be changed such that the user can explore the sequence itself while the PCCA loads. Unfortunately, this could prove to be difficult since the simplification algorithm relies on the PCCA results to simplify the sequence. If an alternative metric could be implemented to simplify the data-set without the use of PCCA, then the progressive streaming of the PCCA could be implemented.

A minor improvement that was suggested during the case study was the fact that nothing in the UI alerted the user to the fact that the optimal clustering was already rendered - this led to the analyst wasting time coming to the same conclusion the system already had.

The dominant state filter needs improvement as well, as the current implementation does not take into account the fact that the dominant state of a cluster may not occur throughout all of it, leaving many states without any transitions to / from the dominant state. This leads the filter to render them with 0 opacity, making them invisible, thereby leaving the filter with limited usefulness.

7.1.2 Back-end

NeoMD can only handle systems with one atom type - this is a limitation that will be addressed in the near future, as it is critical for analysts to be able to process nano-particles that are alloys of various metals. This missing feature is especially apparent when ASE analyses are being run - other than that, the visualizations will

work on any arbitrary particle. Adding support for multiple atom types would also open the door for NeoMD to be used for analysis in other domains, such as biology.

Since states can be labeled differently and yet contain many of the same properties, a better way to highlight similar states needs to be developed, either by aggregating similar states into one "canonical" version, or by changing the way states are highlighted to include states that are similar. This would also mean improving the similarity algorithms in use by the system, and perhaps developing a way to identify states that are similar before the visualization is even displayed. This could be a way to further optimize the system's speed and reduce the amount of redundant states.

The transition region detection / simplification algorithm needs to be improved to include more states from the cluster being transitioned to. While the fuzzy-membership metric is a powerful one, it is not enough to highlight all of the states within a transition region. Applying machine learning techniques can help classify these transition regions, and perhaps enable scientists to classify reaction pathways; no such system exists as of now, as transition regions are highly variable. However, such an approach may prove to be difficult, as it has not been proven that transitions occur similarly in all types of particles. Despite its flaws, the current detection algorithm brings the analyst's attention to the states near the region of interest.

Larger data-sets (over 15GB+) are yet to be tackled in my system, as even loading the entire sequence into Python leads to out-of-memory errors. This can be remedied by re-writing the critical parts of the code in more performant, low-level languages such as C / C++ or Rust.

Providing a feature that could color trajectories based on clustering results from other trajectories could potentially be very useful for analysts when comparing two or more trajectories. Re-coloring based on clustering could highlight segments of the trajectory where similar states are observed, making it abundantly clear how

two trajectories can be characterized in terms of each other. More work can also be done in discovering new clustering methods based on other hidden characteristics of transition matrices, besides relying on eigenvectors / Schur vectors. One suggestion was to use a joint-diagonalization technique, where the eigenvectors of all matrices are calculated at the same time to find clusters in common between multiple trajectories.

7.2 Conclusion

In this work, I built a visual analytics system that enables the efficient analysis of molecular dynamics simulations that have not been explored in depth due to the heterogeneous energy barrier problem. The heterogeneous energy barrier problem makes it difficult to explore certain kinds of molecular systems with simulation techniques commonly in use today; however, the introduction of ParSplice made it possible to explore these reactions. The system described in this work facilitates the extraction, processing and visualization of the data produced by ParSplice. Each component of the system provides a sub-set of these functionalities - DBExtract is for the extraction of the data, NeoMD processes the data, and NeoMDWeb visualizes the data.

The case study showed that the system works fluidly for ParSplice simulations, leading analysts to insights within minutes. It did so by demonstrating how simple it is determine the characteristic states of a cluster, as well as the locations of transition regions between clusters; additionally, the comparison of trajectories within an ensemble is supported. The system also enables the computation of various calculations that previously were hidden behind complex software packages that required the data to be pre-processed, a time-consuming task that is not worthy of an expert's time; I automated this process to speed up the rate at which MD trajectories can be analyzed.

Analyzing these molecular systems is important, as scientists are attempting to

explain the fundamental behavior of nano-particles, and other classes of systems that have not been explored. Understanding the fundamental behavior of nano-particles will have long-lasting impacts in the applications of these materials in various industries, particularly in catalyzing chemical reactions and reducing pollution emissions. Perhaps more importantly, gaining an understanding of nano-particles will give human-kind yet another perspective on the true nature of our reality, as understanding the microscopic fragments of our universe leads to an understanding of the macro-cosmic - "As above, so below".

REFERENCES

"Color brewer", `http://colorbrewer2.com`, accessed: 2022-04-08 (2003).

"D3", `https://d3js.org/`, accessed: 2022-04-11 (2011).

"Celery", `https://github.com/celery/celery`, accessed: 2022-06-13 (2016).

"FastAPI", `https://fastapi.tiangolo.com`, accessed: 2022-06-13 (2019).

Abello, J., F. van Ham and N. Krishnan, "ASK-GraphView: A large scale graph visualization system", IEEE Transactions on Visualization and Computer Graphics **12**, 5, 669–676 (2006).

Ackland, G. J. and A. P. Jones, "Applications of local crystal structure measures in experiment and simulation", Physical Review B **73**, 5, 054104:1–054104:7 (2006).

Ahrens, J., B. Geveci and C. Law, "ParaView: An end-user tool for large data visualization", Tech. rep., Los Alamos National Laboratory (2002).

Andrienko, N. and G. Andrienko, *Visual Analytics of Vessel Movement*, pp. 149–170 (Springer International Publishing, Cham, 2021).

Balbuena, P. B., S. R. Calvo, R. Callejas-Tovar, Z. Gu, G. E. Ramirez-Caballero, P. Hirunsit and Y. Ma, *Challenges in the Design of Active and Durable Alloy Nanocatalysts for Fuel Cells*, vol. 50 of *Modern Aspects of Electrochemistry*, pp. 351–396 (Springe, New York, NY, USA, 2010).

Bekker, H., H. J. C. Berendsen, E. J. Dijkstra, S. Achterop, R. van Drunen, D. van der Spoel, A. Sijbers, H. Keegstra and M. K. R. Renardus, "Gromacs: A parallel computer for molecular dynamics simulations", in "Physics Computing", edited by R. DeGroot and J. Nadrchal, pp. 252–256 (1993).

Bell, A. T., "The impact of nanoscience on heterogeneous catalysis", Science **299**, 5613, 1688–1691 (2003).

Berendsen, H., "Biophysical applications of molecular dynamics", Computer Physics Communications **44**, 3, 233–242 (1987).

Buering, T., J. Gerken and H. Reiterer, "User interaction with scatterplots on small screens - a comparative evaluation of geometric-semantic zoom and fisheye distortion", IEEE Transactions on Visualization and Computer Graphics **12**, 5, 829–836 (2006).

Chae, J., D. Bhowmik, H. Ma, A. Ramanathan and C. Steed, "Visual analytics for deep embeddings of large scale molecular dynamics simulations", in "Proceedings of the IEEE International Conference on Big Data (Big Data)", pp. 1759–1764 (2019).

Cockburn, A., A. Karlson and B. B. Bederson, "A review of overview+detail, zooming, and focus+context interfaces", ACM Computing Surveys **41**, 1, 2:1–2:31 (2009).

Conti, G., J. Grizzard, M. Ahamad and H. Owen, "Visual exploration of malicious network objects using semantic zoom, interactive encoding and dynamic queries", in "Proceedings of the IEEE Workshop on Visualization for Computer Security", VizSEC 05, p. 83–90 (2005).

Cordes, F., M. Weber and J. Schmidt-Ehrenberg, "Metastable conformations via successive Perron-Cluster Cluster Analysis of dihedrals", Tech. rep., Zuse Institute Berlin (2002).

Craig, P. and J. Kennedy, "Coordinated graph and scatter-plot views for the visual exploration of microarray time-series data", in "Proceedings of the IEEE Symposium on Information Visualization", pp. 173–180 (2003).

Deuflhard, P. and M. Weber, "Robust Perron cluster analysis in conformation dynamics", Linear Algebra and its Applications **398**, 161–184 (2005).

Dodge, Y., *Kolmogorov–Smirnov Test*, pp. 283–287 (Springer, New York, NY, 2008).

Dunsmuir, D., *Semantic Zoom View: A Focus+Context Technique for Visualizing a Document Collection*, Master's thesis, Simon Fraser University, Burnaby, BC, Canada (2011).

Durrant, J. D. and J. A. McCammon, "Molecular dynamics simulations and drug discovery", BMC Biology **9**, 1, 71:1–71:9 (2011).

Elmqvist, N., P. Dragicevic and J.-D. Fekete, "Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation", IEEE Transactions on Visualization and Computer Graphics **14**, 6, 1539–1148 (2008).

Faken, D. and H. Jónsson, "Systematic analysis of local atomic structure combined with 3D computer graphics", Computational Materials Science **2**, 2, 279–286 (1994).

Fekete, J.-D., D. Fisher, A. Nandi and M. Sedlmair, "Progressive data analysis and visualization", Dagstuhl Reports **8**, 10, 1–40 (2019).

Garcia, J., R. Theron and F. Garcia, "Semantic zoom: A details on demand visualisation technique for modelling OWL ontologies", in "Highlights in Practical Applications of Agents and Multiagent Systems", edited by J. B. Pérez, J. M. Corchado, M. N. Moreno, V. Julián, P. Mathieu, J. Canada-Bago, A. Ortega and A. F. Caballero, vol. 89 of *Advances in Intelligent and Soft Computing*, p. 85–92 (2011).

Hamelberg, D., J. Mongan and J. A. McCammon, "Accelerated molecular dynamics: A promising and efficient simulation method for biomolecules", The Journal of chemical physics **120**, 24, 11919–11929 (2004).

Henkelman, G., "Atomistic simulations of activated processes in materials", Annual Review of Materials Research **47**, 1, 199–216 (2017).

Henkelman, G., B. P. Uberuaga and H. Jónsson, "A climbing image nudged elastic band method for finding saddle points and minimum energy paths", The Journal of Chemical Physics **113**, 22, 9901–9904 (2000).

Hildebrand, P. W., A. S. Rose and J. K. Tiemann, "Bringing molecular dynamics simulation data into view", Trends in Biochemical Sciences **44**, 11, 902–913 (2019).

Hospital, A., J. R. Goñi, M. Orozco and J. L. Gelpi, "Molecular dynamics simulations: Advances and applications", Advances and Applications in Bioinformatics and Chemistry **8**, 37–47 (2015).

Huang, R., L.-T. Lo, Y. Wen, A. F. Voter and D. Perez, "Cluster analysis of accelerated molecular dynamics simulations: A case study of the decahedron to icosahedron transition in Pt nanoparticles", The Journal of Chemical Physics **147**, 15, 152717:1–152717:6 (2017).

Huang, R., Y. Wen, A. F. Voter and D. Perez, "Direct observations of shape fluctuation in long-time atomistic simulations of metallic nanoclusters", Physical Review Materials **2**, 126002:1–126002:9 (2018).

Jónsson, H., G. Mills and K. W. Jacobsen, "Nudged elastic band method for finding minimum energy paths of transitions", (1998).

Kincaid, R., "SignalLens: Focus+context applied to electronic time series", IEEE Transactions on Visualization and Computer Graphics **16**, 6, 900–907 (2010).

Larsen, A. H., J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus *et al.*, "The atomic simulation environment—a Python library for working with atoms", Journal of Physics: Condensed Matter **29**, 27, 273002:1–273002:30 (2017).

Liu, H., S. Jin, Y. Yan, Y. Tao and H. Lin, "Visual analytics of taxi trajectory data via topical sub-trajectories", Visual Informatics **3**, 3, 140–149 (2019).

Mazimpaka, J. D. and S. Timpf, "Trajectory data mining: A review of methods and applications", Journal of Spatial Information Science **13**, 1, 61–99 (2016).

McGibbon, R. T., K. A. Beauchamp, M. P. Harrigan, C. Klein, J. M. Swails, C. X. Hernández, C. R. Schwantes, L.-P. Wang, T. J. Lane and V. S. Pande, "MDTraj: a modern open library for the analysis of molecular dynamics trajectories", Biophysical Journal **109**, 8, 1528–1532 (2015).

Michaud-Agrawal, N., E. J. Denning, T. B. Woolf and O. Beckstein, "MDAnalysis: a toolkit for the analysis of molecular dynamics simulations", Journal of Computational Chemistry **32**, 10, 2319–2327 (2011).

Miron, R. A. and K. A. Fichthorn, "Multiple time-scale accelerated molecular dynamics: Addressing the small-barrier problem", Physical Review Letters **93**, 12, 128301:1–128301:4 (2004).

Mixcoha, E., R. Rosende, R. Garcia-Fandino and Á. Piñeiro, "Cyclo-lib: A database of computational molecular dynamics simulations of cyclodextrins", Bioinformatics **32**, 21, 3371–3373 (2016).

Miyazaki, N. and Y. Shiozaki, "Calculation of mechanical properties of solids using molecular dynamics method", JSME international journal. Ser. A, Mechanics and material engineering **39**, 4, 606–612 (1996).

Naeem, M., T. Jamal, J. Diaz-Martinez, S. A. Butt, N. Montesano, M. I. Tariq, E. De-la Hoz-Franco and E. De-La-Hoz-Valdiris, "Trends and future perspective challenges in big data", in "Advances in Intelligent Data Analysis and Applications", pp. 309–325 (Springer, 2022).

Newport, T. D., M. S. P. Sansom and P. J. Stansfeld, "The MemProtMD database: A resource for membrane-embedded protein structures and their lipid interactions", Nucleic Acids Research **47**, D1, D390–D397 (2019).

Paula Afonso, A., A. Ferreira, L. Ferreira and R. Vaz, "Rosetrajvis: Visual analytics of trajectories with rose diagrams", in "Proceedings of the International Conference Information Visualisation", pp. 378–384 (2020).

Perez, D., E. D. Cubuk, A. Waterland, E. Kaxiras and A. F. Voter, "Long-time dynamics through parallel trajectory splicing", Journal of Chemical Theory and Computation **12**, 1, 18–28 (2016).

Priedhorsky, R. and T. Randles, "Charliecloud: Unprivileged containers for user-defined software stacks in hpc", in "Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis", SC '17, pp. 36:1–36:10 (2017).

Reuter, B., M. Weber, K. Fackeldey, S. Röblitz and M. E. Garcia, "Generalized markov state modeling method for nonequilibrium biomolecular dynamics: Exemplified on amyloid β conformational dynamics driven by an oscillating electric field", Journal of Chemical Theory and Computation **14**, 7, 3579–3594 (2018).

Roberts, J. C., "State of the art: Coordinated & multiple views in exploratory visualization", in "Proceedings of the International Conference on Coordinated and Multiple Views in Exploratory Visualization", CMV 2007, p. 61–71 (2007).

Rodrigues, E. M., N. Milic-Frayling, M. Smith, B. Shneiderman and D. Hansen, "Group-in-a-box layout for multi-faceted analysis of communities", in "Proceedings of the IEEE International Conference on Privacy, Security, Risk, and Trust, and IEEE International Conference on Social Computing", pp. 354–361 (2011).

Rodríguez-Espigares, I., M. Torrens-Fontanals, J. K. S. Tiemann, D. Aranda-García, J. M. Ramírez-Anguita, T. M. Stepniewski, N. Worp, A. Varela-Rial, A. Morales-Pastor, B. Medel-Lacruz, G. Pándy-Szekeres, E. Mayol, T. Giorgino, J. Carlsson, X. Deupi, S. Filipek, M. Filizola, J. C. Gómez-Tamayo, A. Gonzalez, H. Gutiérrez-de Terán, M. Jiménez-Rosés, W. Jespers, J. Kapla, G. Khelashvili, P. Kolb, D. Latek, M. Marti-Solano, P. Matricon, M.-T. Matsoukas, P. Miszta, M. Olivella, L. Perez-Benito, D. Provasi, S. Ríos, I. R. Torrecillas, J. Sallander, A. Sztyler, S. Vasile, H. Weinstein, U. Zachariae, P. W. Hildebrand, G. De Fabritiis, F. Sanz, D. E. Gloriam, A. Cordomi, R. Guixà-González and J. Selent, "GPCRmd uncovers the dynamics of the 3D-GPCRome", Nature Methods **17**, 8, 777–787 (2020).

Roduner, E., "Size matters: Why nanomaterials are different", Chemical Society Reviews **35**, 7, 583–592 (2006).

Roe, D. R. and T. E. Cheatham, "PTRAJ and CPPTRAJ: software for processing and analysis of molecular dynamics trajectory data", Journal of Chemical Theory and Computation **9**, 7, 3084–3095 (2013).

Sajanlal, P. R., T. S. Sreeprasad, A. K. Samal and T. Pradeep, "Anisotropic nanomaterials: Structure, growth, assembly, and functions", Nano reviews **2**, 1, 5883:1–5883:62 (2011).

Scherer, M. K., B. Trendelkamp-Schroer, F. Paul, G. Pérez-Hernández, M. Hoffmann, N. Plattner, C. Wehmeyer, J.-H. Prinz and F. Noé, "PyEMMA 2: A software package for estimation, validation, and analysis of markov models", Journal of Chemical Theory and Computation **11**, 11, 5525–5542 (2015).

Schott, J. R., *Matrix Analysis for Statistics*, pp. 177–180 (John Wiley & Sons, Hoboken, New Jersey, USA, 2016).

Seeber, M., M. Cecchini, F. Rao, G. Settanni and A. Caflisch, "Wordom: A program for efficient analysis of molecular dynamics simulations", Bioinformatics **23**, 19, 2625–2627 (2007).

Skånberg, R., M. Linares, C. König, P. Norman, D. Jönsson, I. Hotz and A. Ynnerman, "VIA-MD: Visual interactive analysis of molecular dynamics", in "Proceedings of the Workshop on Molecular Graphics and Visual Analysis of Molecular Data", pp. 19–27 (2018).

Stepanov, A., A. Golubev, S. Nikitin and Y. Osin, "A review on the fabrication and properties of platinum nanoparticles", Reviews On Advanced Materials Science **38**, 2, 160–175 (2014).

Steptoe, M., R. Krüger, R. Garcia, X. Liang and R. Maciejewski, "A visual analytics framework for exploring theme park dynamics", ACM Transactions on Interactive Intelligent Systems **8**, 1, 4:1–4:27 (2018).

Stukowski, A., "Visualization and analysis of atomistic simulation data with OVITO–The open visualization tool", Modelling and Simulation in Materials Science and Engineering **18**, 1, 015012:1–015012:7 (2010).

Thompson, A. P., H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in 't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott and S. J. Plimpton, "LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales", Computer Physics Communications **271**, 108171 (2022).

Tominski, C., H. Schumann, G. Andrienko and N. Andrienko, "Stacking-based visualization of trajectory attribute data", IEEE Transactions on Visualization and Computer Graphics **18**, 12, 2565–2574 (2012).

Truhlar, D. G., B. C. Garrett and S. J. Klippenstein, "Current status of transition-state theory", The Journal of Physical Chemistry **100**, 31, 12771–12800 (1996).

Uddin, M. R., C. Ravishankar and V. J. Tsotras, "Finding regions of interest from trajectory data", in "Proceedings of the IEEE International Conference on Mobile Data Management", vol. 1, pp. 39–48 (2011).

van der Kamp, M. W., R. D. Schaeffer, A. L. Jonsson, A. D. Scouras, A. M. Simms, R. D. Toofanny, N. C. Benson, P. C. Anderson, E. D. Merkley, S. Rysavy, D. Bromley, D. A. Beck and V. Daggett, "Dynameomics: A comprehensive database of protein dynamics", Structure **18**, 4, 423–435 (2010).

Vert, M., Y. Doi, K.-H. Hellwich, M. Hess, P. Hodge, P. Kubisa, M. Rinaudo and F. Schué, "Terminology for biorelated polymers and applications (IUPAC recommendations 2012)", Pure and Applied Chemistry **84**, 2, 377–410 (2012).

Voter, A. F., "Hyperdynamics: Accelerated molecular dynamics of infrequent events", Physical Review Letters **78**, 20, 3908–3911 (1997).

Voter, A. F., "Parallel replica method for dynamics of infrequent events", Physical Review B **57**, 22, R13985–R13988 (1998).

Yoshida, M., J.-R. Kahng, J.-S. Moon, K.-H. Jung, K. Kim, H. Sung, C. Lee, C.-K. Kim, W. Yang, G. Jin and K.-S. Oh, "Three series-connected transistor model for a recess-channel-array transistor and improvement of electrical characteristics by a bottom fin structure", Japanese Journal of Applied Physics **48**, 5, 054504:1–054504:4 (2009).

Zhang, Y., W. Luo, E. A. Mack and R. Maciejewski, "Visualizing the impact of geographical variations on multivariate clustering", Computer Graphics Forum **35**, 3, 101–110 (2016).