8-2019

High Performance Computing Techniques to Better Understand Protein Conformational Space

Arpita Joshi
University of Massachusetts Boston

HIGH PERFORMANCE COMPUTING TECHNIQUES TO BETTER UNDERSTAND PROTEIN CONFORMATIONAL SPACE

A Dissertation Presented
by
ARPITA JOSHI

Submitted to the Office of Graduate Studies,
University of Massachusetts Boston,
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY
August 2019

Computer Science Department

HIGH PERFORMANCE COMPUTING TECHNIQUES TO BETTER UNDERSTAND PROTEIN CONFORMATIONAL SPACE

A Dissertation Presented
by
ARPITA JOSHI

Approved as to style and content by:

Nurit Haspel, Associate Professor
Chairperson of the Committee

Eduardo Gonzalez, Associate Professor
Member

Ouyang Ming, Associate Professor
Member

Dan Simovici, Professor
Member

Dan Simovici, Graduate Program Director
Computer Science Department

Marc Pomplun, Chair
Computer Science Department

ABSTRACT

HIGH PERFORMANCE COMPUTING TECHNIQUES TO BETTER UNDERSTAND PROTEIN CONFORMATIONAL SPACE

August 2019

Arpita Joshi, B.E., Rajiv Gandhi Technical University, India
M.S., University of Massachusetts Boston
PhD., University of Massachusetts Boston

Directed By Professor Nurit Haspel

This thesis presents an amalgamation of high performance computing techniques to get better insight into protein molecular dynamics. Key aspects of protein function and dynamics can be learned from their conformational space. Datasets that represent the complex nuances of a protein molecule are high dimensional. Efficient dimensionality reduction becomes indispensable for the analysis of such exorbitant datasets. Dimensionality reduction forms a formidable portion of this work and its application has been explored for other datasets as well. It begins with the parallelization of a known non-liner feature reduction algorithm called Isomap. The code for the algorithm was re-written in C with portions of it parallelized using OpenMP. Next, a novel data instance reduction method was devised which evaluates the information content offered by each data point, which ultimately helps in truncation of the dataset with much fewer data points to evaluate. Once a framework has been established to reduce the number of variables representing a dataset, the work is extended to explore algebraic topology techniques to extract meaningful information from these datasets. This step is the one that helps in sampling the conformations of interest of a protein molecule. The method employs the notion of hierarchical clustering to identify classes within a molecule, thereafter, algebraic topology is used to analyze these classes. Finally, the work is concluded by presenting an approach to solve the open problem of protein folding. A Monte-Carlo based tree search algorithm is put forth to simulate the pathway that a certain protein conformation undertakes to reach another conformation.

The dissertation, in its entirety, offers solutions to a few problems that hinder the progress of solution for the vast problem of understanding protein dynamics. The motion of a protein molecule is guided by changes in its energy profile. In this course the molecule gradually slips from one energy class to another. Structurally, this switch is transient spanning over milliseconds or less and hence is difficult to be captured solely by the work in wet laboratories.

ACKNOWLEDGMENTS

I thank my dad, Dr. B.K.Joshi, my mom Mrs. Kshama Joshi, my brother Prakhar, my PhD adviser Dr. Nurit Haspel, my thesis committee members and mentors Dr. Dan Simovici, Dr. Eduardo Gonazalez, Dr. Ming Ouyang and Dr. Peter Fejer. I also thank the following people who are close to me, Jinu Thomas, Shreejaya Dutt and Rajesh Uniyal.

TABLE OF CONTENTS

CHAPTER Page

LIST OF FIGURES

Figure Page

LIST OF TABLES

Chapter 1

INTRODUCTION

High Performance Computing (HPC) is defined as the process of amassing the power of a *supercomputer* efficiently in order to reduce the computation time on large data sets. A protein molecule's representation often needs data of the order of 25 MB. Efficient HPC techniques can be a boon in processing and understanding data of such magnitude. This is the basis for the work here.

1.1 Machine Architectures

John von Neumann first wrote the general requirements for an electronic computer in 1945. Earlier, the computers were programmed through *hard wiring*. But over the years, the notion of *stored-program computer* has evolved. Now, both the program instructions and the data are kept in electronic memory. Processor is a logic circuitry that responds to and processes the basic instructions that drive a computer. Central to a processor is an ALU (Arithmetic and Logic Unit). A processor has its own memory inside it in the shape of small cells. Each memory cell is called a *register*. Registers are used to store data temporarily for performing operations. Control Unit, is responsible for all the activities of the processor and also manages the input and output devices of the computer. It also controls the flow of instructions, obtains instructions from the program stored in main memory, interprets (translation of instructions into computer language) the instructions, and issues signals that cause other units of the computer to execute them.[1]

A single core processor has only one processor, so it can only start one operation at a time. It can however in some situations start a new operation before the previous one is complete[1]. A multi-core processor is a processing system composed of two or more independent cores. It is basically an integrated circuit to which two or more individual processors have been attached. Multi-core machines have various advantages like better resource utilization, efficient data sharing, increased performance etc.

[1]One example, process scheduling might invoke a process with higher priority

1.1.1 Flynn's Taxonomy

Following are excerpts from [2].

- **SISD:**
  Short for Single Instruction Stream, Single Data Stream. It is a serial computer that acts on only one instruction during any one clock cycle. Also, only one data stream is being used as input during any one clock cycle. It has a deterministic execution.

- **SIMD:**
  It is a contraction for Single Instruction Stream, Multiple Data Stream. It is a type of parallel computers. All processing units in this type of machine execute the same instruction at any given clock cycle, but it differs from an SISD machine because each of the processing units here can operate on different data sets. Such machines are best suited for specialized problems characterized by a high degree of regularity, such as graphics and image processing. These machines are synchronous (lockstep) and have a deterministic execution. Most modern computers, particularly those with graphics processor units (GPUs) employ SIMD instruction and execution units.

- **MISD:**
  Short for Multiple Instruction Stream, Single Data Stream are another type of parallel computers. In this case, each processing unit operates on the data independently via separate instruction streams. A single data stream is fed into multiple processing units. Very few examples of such a scenario have ever existed. Some conceivable use cases could be, multiple frequency filters operating on a single signal stream and multiple cryptography algorithms attempting to crack a single coded message.

- **MIMD:**
  It is a contraction for Multiple Instruction Stream, Multiple Data Stream and is the most common type of parallel computers around. Here, every processor might be executing a different instructions stream and working on a different data stream. Execution could be both synchronous or asynchronous and deterministic or non-deterministic. Many MIMD architectures also include SIMD execution subcomponents.

1.1.2 Graphics Processing Unit

A GPU was originally designed to meet the growing needs of 3D gaming industry, but eventually it developed into a more general high-performance computing device. As is clear from Figure-1.1 a GPU has multiple control units. Each of which has its own cache. A separate cache for each control unit is key to the speed-up offered by algorithms implemented over GPUs. In turn, each control unit has its own series of ALUs. These

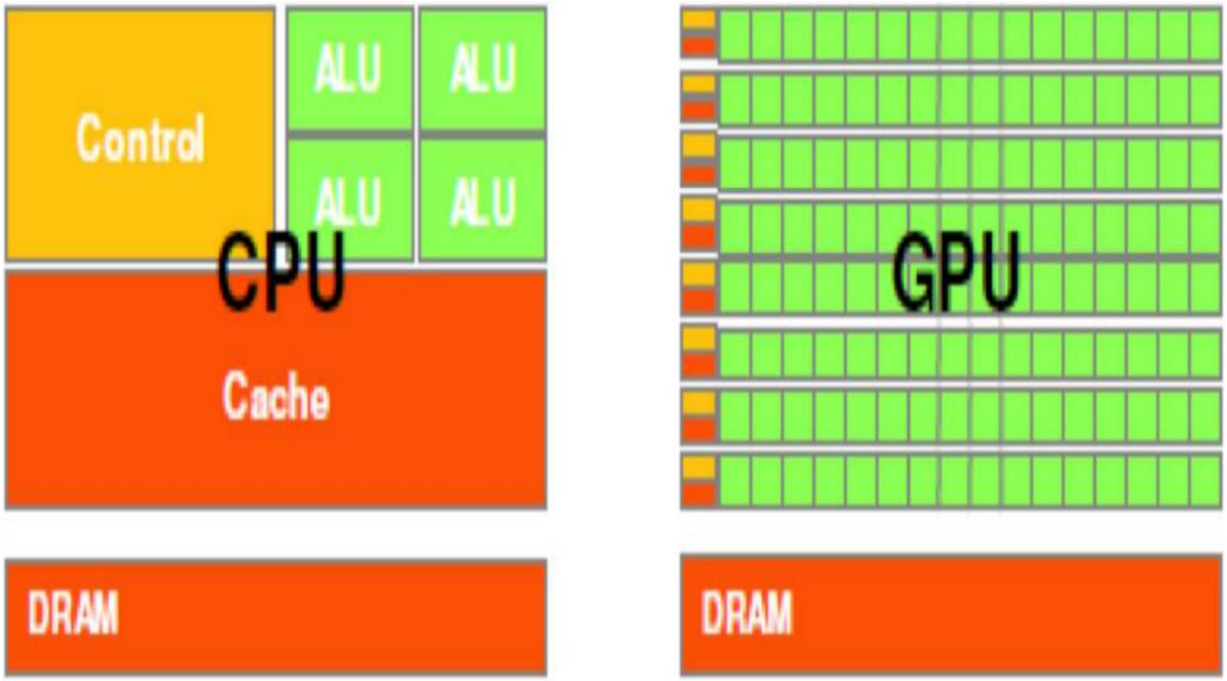


Figure 1.1: Fundamental Differences between CPU and GPU

separate ALUs can function independent of each other and can tackle different portions of execution simultaneously.

1.2 Parallel Computing

The foremost step in parallelizing a piece of code is the optimization of serial code. Sub-optimal serial algorithms can result in inefficient and unreliable speed-up. Following is the breakdown of the process at a coarse level:

- Identify portions of the work that can be parallelized.
- Mapping the concurrent pieces of work onto multiple processes running in parallel
- Distributing the input, output and intermediate data associated with the program
- Managing accesses to data shared by multiple processors
- Synchronizing the processors at various stages of the parallel program execution

The process of dividing a computation into smaller parts, some or all of which may potentially be executed in parallel, is called *decomposition*. *Tasks* are programmer defined units of computation into which the main computation is subdivided by means of decomposition. Some tasks may use data produced by other tasks and thus may need to wait for these tasks to finish execution. An abstraction used to express such dependencies among tasks and their relative order of execution is known as a *task-dependency graph*.

The number and size of tasks into which a problem is decomposed determines the *granularity* of the decomposition. A decomposition into a large number of small tasks is called *fine-grained* and a decomposition into a small number of large tasks is called *coarse-grained*. Maximum number of tasks that can be executed simultaneously in a parallel program at any given time is known as its *maximum degree of concurrency*. Expectedly, the *average degree of concurrency*, is the average number of tasks that can run concurrently over the entire duration of execution of the program.

As mentioned earlier, one of the fundamental steps that we need to undertake to solve a problem in parallel is to split the computations to be performed. Following are the ways in which we can to do so-

- Recursive Decomposition is a method for inducing concurrency in problems that can be solved using the divide-and-conquer strategy. In this technique, a problem is solved by first dividing it into a set of independent subproblems. Each one of these subproblems is solved by recursively applying a similar division into smaller subproblems followed by a combination of their results.The divide-and-conquer strategy results in natural concurrency, as different subproblems can be solved concurrently.

- Data decomposition is a powerful and commonly used method for deriving concurrency in algorithms that operate on large data structures. In this method, the decomposition of computations is done in two steps. In the first step, the data on which the computations are performed is partitioned, and in the second step, this data partitioning is used to induce a partitioning of the computations into tasks.

- Exploratory decomposition is used to decompose problems whose underlying computations correspond to a search of a space for solutions. In exploratory decomposition, we partition the search space into smaller parts, and search each one of these parts concurrently, until the desired solutions are found.

- Speculative decomposition is used when a program may take one of many possible computationally significant branches depending on the output of other computations that precede it. In this situation, while one task is performing the computation whose output is used in deciding the next computation, other tasks can concurrently start the computations of the next stage.

The serial runtime of a program is the time elapsed between the beginning and the end of its execution on a sequential computer. The *parallel runtime* is the time that elapses from the moment a parallel computation starts to the moment the last processing element finishes execution. Following are the metrics that can be used to guage the efficiency of a parallelized program-

- The overheads incurred by a parallel program are encapsulated into a single expression referred to as the *overhead function*. We define overhead function or

total overhead of a parallel system as the total time collectively spent by all the processing elements over and above that required by the fastest known sequential algorithm for solving the same problem on a single processing element.

- When evaluating a parallel system, we are often interested in knowing how much performance gain is achieved by parallelizing a given application over a sequential implementation. Speedup is a measure that captures the relative benefit of solving a problem in parallel. It is defined as the ratio of the time taken to solve a problem on a single processing element to the time required to solve the same problem on a parallel computer.

- *Efficiency* is a measure of the fraction of time for which a processing element is usefully employed; it is defined as the ratio of speedup to the number of processing elements.

- We define the *cost* of solving a problem on a parallel system as the product of parallel runtime and the number of processing elements used. Cost reflects the sum of the time that each processing element spends solving the problem.

1.3 Parallel Programming Platforms

Numerous programming languages and libraries have been developed for explicit parallel programming. These differ in their view of the address space that they make available to the programmer, the degree of synchronization imposed on concurrent activities, and the multiplicity of programs. Here we explore three such platforms.

1.3.1 Message-Passing Paradigm

The message-passing programming paradigm is one of the oldest and most widely used approaches for programming parallel computers. Message-passing programs are often written using the asynchronous or loosely synchronous paradigms. In the asynchronous paradigm, all concurrent tasks execute asynchronously. This makes it possible to implement any parallel algorithm. However, such programs can be harder to reason about, and can have non-deterministic behavior due to race conditions. In loosely synchronous programs, tasks or subsets of tasks synchronize to perform interactions. However, between these interactions, tasks execute completely asynchronously. Since the interaction happens synchronously, it is still quite easy to reason about the program.

MPI: the Message Passing Interface

MPI defines a standard library for message-passing that can be used to develop portable message-passing programs using either C or Fortran. The MPI standard defines both the syntax as well as the semantics of a core set of library routines that are very useful in writing message-passing programs. MPI was developed by

a group of researchers from academia and industry, and has enjoyed wide support by almost all the hardware vendors. The MPI library contains over 125 routines, but the number of key concepts is much smaller. In fact, it is possible to write fully-functional message-passing programs by using only the six routines shown in Table-1.1.

Table 1.1: Main MPI routines.

Routine	**Description**
MPI_Init	Initializes MPI
MPI_Finalize	Terminates MPI
MPI_Comm_size	Determines the number pf processes
MPI_Comm_rank	Determines the label of the calling process
MPI_Send	Sends a message
MPI_Recv	Receives a message

The binding and calling sequences of MPI_Init and MPI_Finalize functions are illustrative of the naming practices and argument conventions followed by MPI. All MPI routines, data-types, and constants are prefixed by "MPI_ ". The return code for successful completion is MPI_SUCCESS .This and other MPI constants and data-structures are defined for C in the file "mpi.h" . This header file must be included in each MPI program.

1.3.2 Shared Address Space Paradigm

Explicit parallel programming requires specification of parallel tasks along with their interactions. These interactions may be in the form of synchronization between concurrent tasks or communication of intermediate results. In shared address space architectures,communication is implicitly specified since some (or all) of the memory is accessible to all the processors. Consequently, programming paradigms for shared address space machines focus on constructs for expressing concurrency and synchronization along with techniques for minimizing associated overheads. Shared address space programming paradigms can vary on mechanisms for data sharing,concurrency models, and support for synchronization.

The overheads associated with enforcing protection domains make processes less suitable for parallel programming. In contrast, lightweight processes and threads assume that all memory is global. By relaxing the protection domain, lightweight processes and threads support much faster manipulation. Directive based programming models extend the threaded model by facilitating creation and synchronization of threads. A *thread* is a single stream of control in the flow of a program. Threaded programming models offer significant advantages over message-passing programming models as under:

Software Portability: Threaded applications can be developed on serial machines and run on parallel machines without any changes.

Latency Hiding: One of the major overheads in programs (both serial and parallel) is the access latency for memory access, I/O, and communication. By allowing multiple threads to execute on the same processor, threaded APIs enable this latency to be hidden.

Scheduling and Load Balancing: While writing shared address space parallel programs, a programmer must express concurrency in a way that minimizes overheads of remote interaction and idling. Threaded APIs allow the programmer to specify a large number of concurrent tasks and support system-level dynamic mapping of tasks to processors with a view to minimizing idling overheads.

Ease of Programming, Widespread Use: Due to the aforementioned advantages, threaded programs are significantly easier to write than corresponding programs using message passing APIs. Achieving identical levels of performance for the two programs may require additional effort, however. With widespread acceptance of the POSIX thread API, development tools for POSIX threads are more widely available and stable.

POSIX Threads

Also referred to as Pthreads, POSIX has emerged as the standard threads API, supported by most vendors. Multi-threading concepts are largely independent of the API and can be used for programming with other thread APIs (NT threads, Solaris threads, Java threads,etc.) as well. Threads can be created in the Pthreads API using the function *pthread_create*. The prototype of this function is:

```
#include <pthread.h>
int
pthread_create (
pthread_t   *thread_handle,
const pthread_attr_t   *attribute,
void *   (*thread_function)(void*),
void   *arg);
```

The *pthread_create* function creates a single thread that corresponds to the invocation of the function *thread_function* (and any other functions called by *thread_function*). On successful creation of a thread, a unique identifier is associated with the thread and assigned to the location pointed to by t*hread_handle*.

The thread has the attributes described by the attribute argument. When this argument is NULL, a thread with default attributes is created. The arg field specifies a pointer to the argument to function *thread_function*. This argument is typically used to pass the workspace and other thread-specific data to a thread. The *thread_handle* variable is written before the the function *pthread_create* returns; and the new thread is ready for execution as soon as it is created. If the thread is scheduled on the same processor, the new thread may, in fact, preempt its creator.

Using *pthread_create* and *pthread_join* calls, we can create concurrent tasks. These tasks work together to manipulate data and accomplish a given task. When multiple threads attempt to manipulate the same data item, the results can often be incoherent if proper care is not taken to synchronize them. Threaded APIs provide support for implementing critical sections and atomic operations using ***mutex-locks*** *(mutual exclusion locks)*. Mutex-locks have two states: locked and unlocked. At any point of time, only one thread can lock a mutex lock. A lock is an atomic operation generally associated with a piece of code that manipulates shared data. To access the shared data, a thread must first try to acquire a mutex-lock. If the mutex-lock is already locked, the process trying to acquire the lock is blocked. This is because a locked mutex-lock implies that there is another thread currently in the critical section and that no other thread must be allowed in. When a thread leaves a critical section, it must unlock the mutex-lock so that other threads can enter the critical section. All mutex-locks must be initialized to the unlocked state at the beginning of the program.

OpenMP

Conventional wisdom indicates that a large class of applications can be efficiently supported by higher level constructs (or directives) which rid the programmer of the mechanics of manipulating threads. Such directive-based languages have existed for a long time, but only recently have standardization efforts succeeded in the form of OpenMP. OpenMP is an API that can be used with FORTRAN, C, and C++ for programming shared address space machines. OpenMP directives provide support for concurrency,synchronization, and data handling while obviating the need for explicitly setting up mutexes, condition variables, data scope, and initialization. A huge portion of the Parallelized version of Isomap in Chapter-2 has been done using OpenMP. OpenMP directives in C and C++ are based on the *#pragma* compiler directives. The directive itself consists of a directive name followed by clauses.

```
#pragma omp directive [clause list]
```

OpenMP programs execute serially until they encounter the *parallel* directive.

This directive is responsible for creating a group of threads. The exact number of threads can be specified in the directive, set using an environment variable, or at runtime using OpenMP functions. The main thread that encounters the *parallel* directive becomes the master of this group of threads and is assigned the thread id 0 within the group. The parallel directive has the following prototype:

```
pragma omp parallel [clause list]
/* structured block */
```

Each thread created by this directive executes the structured block specified by the parallel directive. The clause list is used to specify conditional parallelization, number of threads, and data handling.

- Conditional Parallelization: The clause *if (scalar expression)* determines whether the parallel construct results in creation of threads. Only one *if* clause can be used with a parallel directive.
- Degree of Concurrency: The clause *num_threads (integer expression)* specifies the number of threads that are created by the parallel directive.
- Data Handling: The clause *private(variable list)* indicates that the set of variables specified is local to each thread – i.e., each thread has its own copy of each variable in the list. The clause *firstprivate(variable list)* is similar to the *private* clause, except the values of variables on entering the threads are initialized to corresponding values before the parallel directive. The clause *shared(variable list)* indicates that all variables in the list are shared across all the threads, i.e., there is only one copy. Special care must be taken while handling these variables by threads to ensure serializability.

1.3.3 Compute Unified Device Architecture (CUDA)

It is an extension of the C programming language and was created by nVidia. Using CUDA allows the programmer to take advantage of the massive parallel computing power of an nVidia graphics card in order to do general purpose computation. Following are excerpts from the beginners CUDA course on NVIDIA's website.

CPUs like Intel Core 2 Duo and AMD Opteron are good at doing one or two tasks at a time, and doing those tasks very quickly. Graphics cards, on the other hand, are good at doing a massive number tasks at the same time, and doing those tasks relatively quickly. To put this into perspective, suppose you have a 20 inch monitor with a standard resolution of 1,920 x 1200. An nVidia graphics card has the computational ability to calculate the color of 2,304,000 different pixels, many times a second. In order to accomplish this feat, graphics cards use dozens, even hundreds of ALUs. Fortunately, nVidia's ALUs are fully programmable, which enables us to harness an unprecedented amount of computational power into the

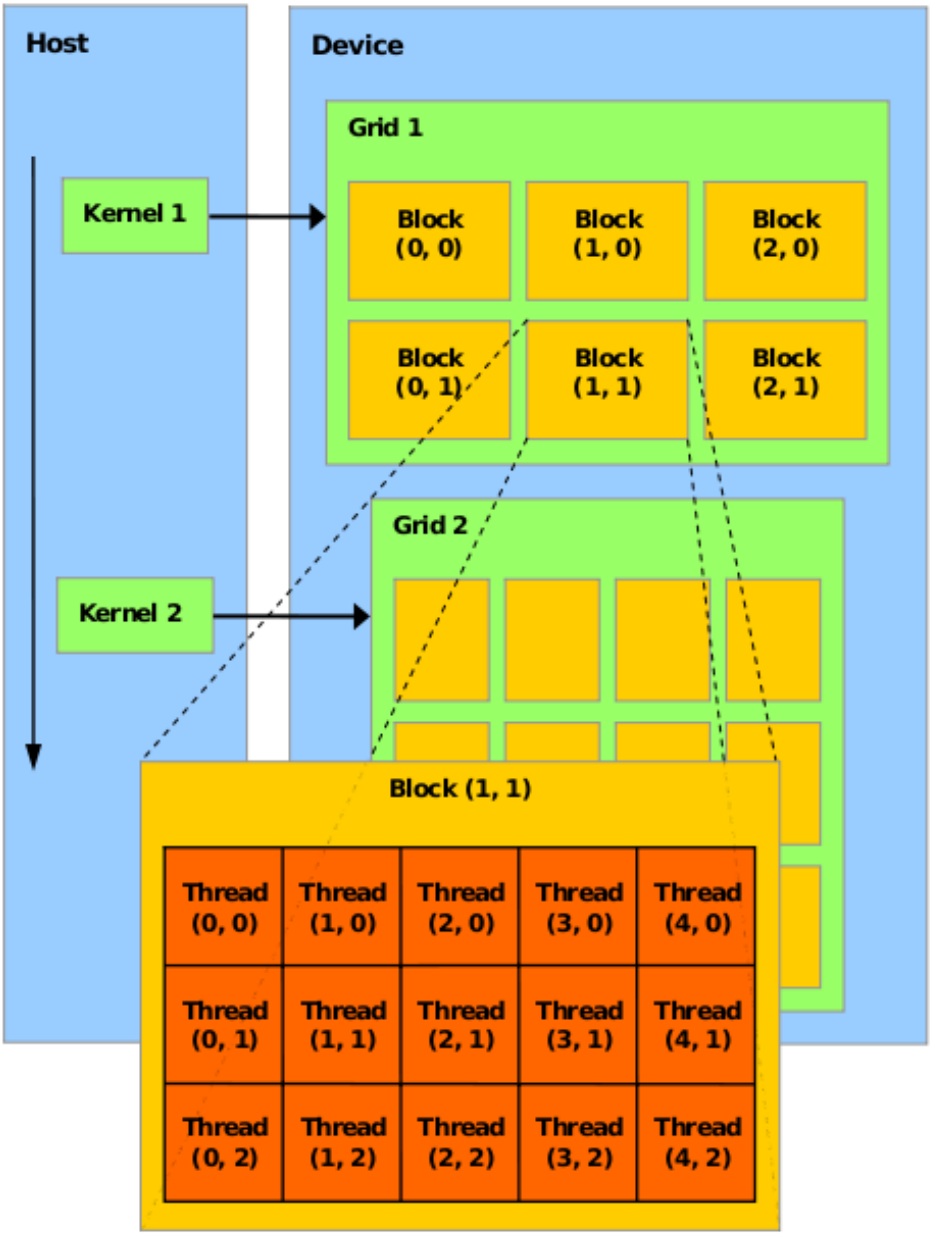


Figure 1.2: CUDA architecture

*As per Nvidia CUDA/C Programming Guide

programs that we write. As stated previously, CUDA lets the programmer take advantage of the hundreds of ALUs inside a graphics processor, which is much more powerful than the handful of ALUs available in any CPU. However, this does put a limit on the types of applications that are well suited to CUDA. A description of the architecture of a GPU was provided in Introduction. In order to run efficiently on a GPU, you need to have many hundreds of threads. Generally, the more threads you have, the better. If you have an algorithm that is mostly serial, then it does not make sense to use CUDA. Many serial algorithms do have parallel equivalents, but many do not. If you can't break your problem down into at least a thousand threads, then CUDA probably is not the best solution for you. If there is one thing that CUDA excels at, it's number crunching. The GPU is fully capable of doing 32-bit integer and floating point operations. In fact, it GPUs are more suited for floating point computations, which makes CUDA an excellent for number crunching. Some of the higher end graphics cards do have double floating point units, however there is only one 64-bit floating point unit for every 16 32-bit floating point units. So using double floating point numbers with CUDA should be avoided if they aren't absolutely necessary for your application. Most modern CPUs have a couple megabytes of L2 cache because most programs have high data coherency. However, when working quickly across a large dataset, say 500 megabytes, the L2 cache may not be as helpful. The memory interface for GPUs is very different from the memory interface of CPUs. GPUs use massive parallel interfaces in order to connect with it's memory. For example, the GTX 280 uses a 512-bit interace to it's high performance GDDR-3 memory. This type of interface is approximately 10 times faster than a typical CPU to memory interface, which is great. It is worth noting that most nVidia graphics cards do not have more than 1 gigabyte of memory. Nvidia does offer special CUDA compute cards which have up to four gigabytes of ram onboard, but these cards are more expensive than cards originally intended for gaming. As stated previously, CUDA can be taken full advantage of when writing in C. This is good news, since most programmers are very familiar with C. Also stated previously, the main idea of CUDA is to have thousands of threads executing in paralle. What wasn't stated is that all of these threads are going to be executing the very same function, known as a kernel. Understanding what the kernel is and how it works is critical to your success when writing an application that uses CUDA. The idea is that even though all of the threads of your program are executing the same function, all of the threads will be working with a different dataset. Each thread will know it's own ID, and based off it's ID, it will determine which pieces of data to work on. Don't worry, flow control like 'if, for, while, do, etc.' are all supported.

One important thing to remember is that your entire program DOES NOT

need to be written in CUDA. If you're writing a large application, complete with a user interface, and many other functions, then most of your code will be written in C++ or whatever your language of choice is. Then, when something extremely computationally intense is needed, your program can simply call the CUDA kernel function you wrote. So the main idea is that CUDA should only be used for the most computationally intense portions of your program. While CUDA is specifically meant to run on nVidia's graphics cards, it can also run on any CPU.

1.4 Protein Folding

Understanding protein structure and dynamics is at the forefront of studies in Bioinformatics and Computational Biology. How and why a protein molecule attains a certain structure is what being attempted to be understood. A protein molecule has a hierarchy of structure. The very first level being the sequence of amino acids it has. The entire sequence of conformational changes that a protein goes through is encoded in this sequence, (Anfinsen's dogma) and this sequence itself is referred to as the primary structure of a protein. The secondary structure of a protein is made up of alpha helices and beta sheets, which are stable folding patterns that result from hydrogen bonding between amino groups and carboxyl groups that are nearby in a protein chain. Interactions between various secondary structure elements like salt bridges, hydrogen bonds, and the tight packing of side chains and disulphide bods lead to tertiary structure. The tertiary structure is referred to as the formulations that occur in a single linear polypeptide chain and lastly, the quaternary structure is the ultimate three dimensional shape attained by a protein molecule with multiple polypeptide chains that operate as a single functional unit (multimer).

A protein when synthesized, attains a final shape which is reached by undergoing through a series of structures. The chosen one is typically the one that is energetically most favorable. Several types of molecular interactions lead up to the folding of a protein. Thermodynamic stability of a molecule plays the biggest role in determining its ultimate structure. Next type of molecular interactions that drive protein motion are the hydrophobic interactions within the molecule. These interactions have their impact even after a protein has found its most stable state, they drive inter-molecular dynamics and *chaperon* proteins (the ones that govern the folding of other proteins). Another type of interaction is the disulfide linkages, they are a sulphur-sulphur chemical bond formed because of oxidation within a protein. [3] Figures 1.3 and 1.4 show the native and folded states of a protein. The inactive form, Trypsinogen when synthesized goes through a series of changes to get activated to ultimately fold into a more stable structure of trypsin.

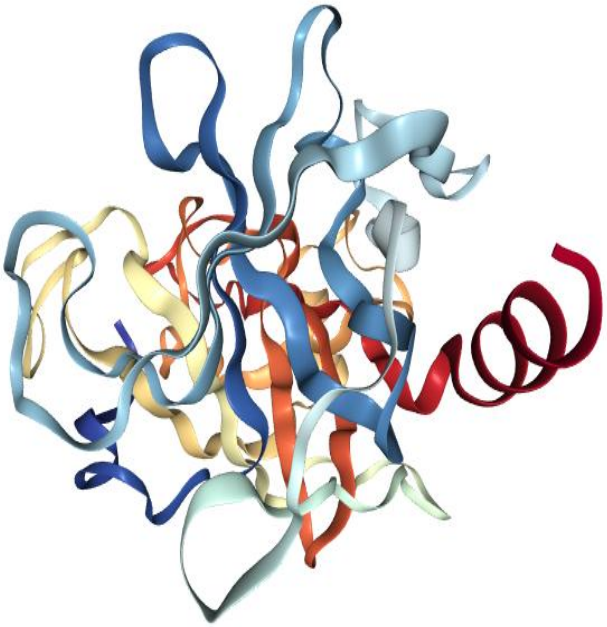


Figure 1.3: Inactive Trypsinogen

Figure 1.4: Active folded form of trypsinogen, Trypsin

*Both the figures above were obtained from www.rcsb.org

1.5 Parallel Computing in Bioinformatics

Data sets representing the complex nature of a macromolecule like protein are often humongous. These data sets are so diverse that storing them itself is a challenge. Computational time of such high dimensional data poses a big problem in understanding it. These data sets can be classified as *'biological big data'*. Expectedly, a wide variety of high performance computing techniques can be employed to deal with it. Parallel programming methods have found application in emulating all the possible pathways of a protein molecule simultaneously. The suitable ones can be extracted using various filtering methods and the synchronization offered by these parallel platforms. Chapter-2 presents one such application.

Bioinformatics is a multidisciplinary field that is in constant evolution due to technological advances in correlated sciences (eg, computer science, biology, mathematical, chemistry, and medicine). Thus, it requires skills from these domains of sciences for modeling, gathering, storing, manipulating, analyzing, and interpreting biological information, ie, "biological big data". Since biological big data is generated by several different bioinformatics/biological/biomedical experiments, it can be presented as structured or unstructured data. Due to the complexity in the nature of the biological big data, a shift to discovery-driven data science is under way, especially in the genomic field. [4]

1.6 Objectives

This thesis is a result of a collection of work each of which was aimed at coming up with way to better understand the structure and dynamic of proteins, all of which can be summed up as under:

1. Write parallelized versions of well-established dimensionality reduction methods best suited for protein data.
2. Design novel algorithms for dimensionality reduction, that are more effective and faster.
3. Use the techniques above to model physical pathways that a protein undertakes.

1.7 Organization of the Thesis

- Chapter 2 describes the parallelization of a known non-linear feature reduction algorithm.
- Chapter 3 proposes and analyses a novel data instance reduction technique.

- Chapter 4 combines the findings of Chapters 2 and 3 to investigate intermediate conformations.
- Chapter 5 takes the results of other chapters to develop an approach to simulate actual protein motion.
- Chapter 6 summarizes the results of all of the research projects and discusses how these can be taken further.

Chapter 2

PARALLELIZATION OF ISOMAP

Ascertaining the conformational landscape of a macromolecule, like protein is indispensable to understanding its characteristics and functions. In this work, an amassment of these techniques are presented, that would be an aid in sampling of these conformations better and faster. The datasets that represent these conformational dynamics of proteins are complex and high dimensional. Therefore, there arises a need for dimensionality reduction methods that best conserve the variance and further the analysis of the data. We present a parallelized version of a well-known dimensionality reduction method, Isomap. Isomap has been shown to produce better results than linear dimensionality reduction in approximating the complex landscape of protein folding. However, the algorithm is compute-intensive for large proteins or a large number of samples, used to model a path that a protein undergoes. We worked on an algorithm, parallelized using OpenMP, with a speed-up of approximately twice. The results are in agreement with the ones obtained using sequential Isomap.

2.1 Representation of Data: The Problem

In order to understand how proteins perform their function, it is essential to characterize their conformational space. Understanding the connection between protein structure, dynamics and function can contribute to our understanding of cellular processes. The question of how the structure and dynamics of proteins relate to their function has challenged scientists for several decades but still remains open. Methods that explore the conformational landscape of proteins include Molecular Dynamics (MD) [5], Monte Carlo sampling [6] geometric-based sampling [7, 8, 9, 10], Elastic Network Modeling [11, 12], normal mode analysis [13, 14], morphing [15] and several other methods. The complex, high dimensional nature of the protein conformational space requires the generation of tens of thousands,

if not more, conformations per trajectory. An average protein contains several hundreds to several thousands of atoms. Therefore, this is an enormous amount of data which requires a large amount of time and space to process, store and analyze. The data is presented in a way that each row represents a different conformation. If a protein has N atoms, then every row that represents the atomic coordinates of the protein has $N \times 3$ variables. All the numbers in the columns together make up for also the orientation that the specific conformation would have. However, due to the mutual constraints between atoms in the protein, the *real* dimensionality of the conformational space is much lower than that. Therefore, one of our preliminary tasks is to find a more efficient way to represent the data. Dimensionality reduction techniques are often used to form a lower-dimensional representation of high dimensional data.

2.2 Other methods

Linear dimensionality reduction like Principal Component Analysis (PCA) and its variants may not capture the complex, non-linear nature of protein conformational landscape. Dimensionality reduction techniques are broadly classified based on the solution space they generate, as convex and non-convex [16]. Techniques described in [17] give explicit details of the various well established non-convex methods. These methods are further sub-divided into Full Spectral Techniques, the ones that perform the eigen decomposition of a full matrix and Sparse Spectral Techniques, the ones that do the same for a sparse matrix. The latter ones have better time-complexity but these approaches are *local*. They attempt at retaining only the local structure that the sparse portions of the dataset present. On the other hand, Full Spectral Techniques, capture the covariance between all the data instances and form a more thorough representation of the structure as a whole. The Isomap algorithm [18] is a non-linear dimensionality reduction method that falls into the Convex Full Spectral category. It takes as input the distances between points in a high-dimensional observation space, and outputs their coordinates in a low-dimensional embedding that best preserves their intrinsic geodesic distances. The algorithm operates in three modes, each of which differs in the kind of data they accept (see Methods). Despite its advantage in efficient representation of molecular data[19, 20], Isomap is computationally expensive, especially with very large, multi-dimensional datasets. To overcome this, we have implemented our version of the Mode-III of Isomap. Improvements over Isomap are presented in [21, 22]. A similar approach is adopted but in a way that is more suited for protein data.

Our method uses a distance function to calculate the distance between the

points that in turn measures the similarity between the conformations that each of these points represent. The algorithm runs twice as fast as the serial implementation with comparable results. The output is a lower-dimensional projection that can be used later for purposes of visualization and analysis. In particular, one of our goals is to detect intermediate structures by obtaining distinct clusters using Persistent Homology as in our previous work [23].

2.3 Isomap and Tweaks to it

The parallelization of the dimensionality reduction algorithm used here, Isomap, begins with the very first step itself of Isomap,

1. The search for k-nearest neighbors of every point in the dataset. This yields a neighborhood graph. If the input matrix is $N \times M$, the neighborhood graph would be $N \times k$ which is made to be $N \times N$ substituting the value of distance for the k neighbors of each point and infinity for the rest (except for the point itself, because distance of a point with itself would be zero).

2. Next comes the computation of the shortest path tree of this graph.

3. Once this is done, Multi-dimensional scaling is performed. The dimension of the matrix still remains $N \times N$.

4. Next, the eigen vectors and values are computed, and the matrix dimension after it, is reduced to $N \times i$, where i is the number of dimensions desired for the final embedding.

Each of these algorithms, is elucidated in detail as under:

- KNN:
  The dataset here is of the order of tens of thousands points. The word 'near' finds its meaning in quantization of the features for every point with respect to every other point. A distance function [18] has been formulated to achieve this. In contrast with the sequential Isomap, each of the data points can be operated on simultaneously. The number of neighbors to be sought for each point are so chosen, so as to obtain one connected component in the resultant neighborhood graph. We opt for least such number.

- Floyd-Warshall:
  Floyd-Warshall's all pair shortest path algorithm can be used for construction of the shortest path tree in case the graph is directed and non-symmetric and

preserves the sense of direction by negative distances. The algorithm has a triple nested loop, of which the middle one executes independently and hence has been parallelized [24]. This parallelization of the middle loop renders the run-time of the algorithm $O(N^2)$.

```
for(int a=0; a<N; a++)
{
#pragma omp parallel for
for(int b=0; b<N; b++)
for(int c=0; c<N; c++)
if (...)
...
}
```

- Dijkstra's Algorithm:
  For the protein data in this work, we have an undirected, symmetric graph with positive distances. So, Dijkstra's algorithm with multiple sources is used. We chose this algorithm over Floyd Warshall's to exploit the symmetry of the neighborhood graph; which allows for having to save only a triangular matrix in the memory, reducing the storage requirements to half. Also, it allows for an early notification if the numbers of neighbors are not enough to get one connected component. In this scenario, the algorithm is allowed to halt at the first iteration itself, knowing that there is at least one point that isn't reachable from the first. At each iteration the number of nodes to be worked with is reduced by one. The first iteration of the loop establishes the shortest paths between the first source node and the rest of the nodes. Subsequent iterations do the same for the remaining nodes. So, by the time the last node of the graph is reached, there is just one node left to work with and that is the node in question itself and so the algorithm walks out of the loop.

- Union-Find:
  The number of connected components in the graph so produced can be found using the classic union-find operations [25]. This portion of code comes into play if the Floyd-Warshall is used as the shortest path tree finding algorithm. While using Dijkstra's algorithm, the first iteration itself tells whether the chosen number of neighbors are enough to find one connected component, if they aren't the program is designed to halt prompting for more neighbors. The largest component found first is chosen to embed. Since we aim at only reducing the dimensionality of the available data in this step, we find as

Table 2.1: Details about data samples

Molecule	Name	No. of atoms	Rows[1]	Columns[2]	k[3]	RMSD error
4did	CDC42, complexed with GDP	1880	20000	534	3	432.98
Hgalanin	Human Galanin, a neuropeptide	434	22750	363	18	532.76
Pgalanin	Porcine Galanin	442	22750	351	8	489.62
Neuromedlin	a mammalian peptide	116	22750	123	7	312.45
Oxytocin	a hormone	135	22750	111	8	328.98
Vasopressin	a hormone	140	22500	111	6	516.23

Table 2.2: Isomap trends in residual variance with increasing dimensions

Molecule	1	2	3	4	Time (min)
4did	0.012	0.006	0.005	0.003	195
Hgalanin	0.544	0.383	0.269	0.184	380
Pgalanin	0.234	0.142	0.138	0.095	362
Neuromedlin	0.665	0.544	0.428	0.349	245
Oxytocin	0.448	0.3	0.199	0.144	253
Vasopressin	0.533	0.246	0.184	0.144	282

many neighbors of each point as would be required to obtain one connected component. This gives a thorough coverage of the conformational space.

- MDS:
  Next, the classical version of Multi-Dimensional Scaling is performed on the data thus far [18].

- Eigenvectors and Eigenalues:
  To obtain a rigid transformation, eigenvectors and values are to be found in as many dimensions as one wishes to observe. The power method is used to find the dominant eigenvalue and its corresponding eigenvector in a loop that stretches for as many iterations as the number of dimensions. The number of dimensions has been set to five to observe the residual variance, and the first three dimensions are embedded and written to a file.

2.4 Analysis of Results

The proteins we used here range between 9 and around 200 amino acids. Each data point here, represents a conformation of the molecule generated using Molecular Dynamics simulations. We typically use an average of twenty two thousand such conformations. Table-1 gives more details about the molecules used. As mentioned

[1]no. of conformations used
[2]no. of features used to represent one conformation of the molecule
[3]no. of neighbors that produced one connected component

earlier, OpenMP has been employed to parallelize the algorithms rewritten in C. Due to the need of computing all pair shortest paths, the time complexity of Isomap is O(N^3). In our version, because we harness the capabilities of a modern multi-core CPU, it has been brought down to $O(N^2)$ when the Floyd Warshall method of shortest path evaluation is used and $O(N^2 logN)$ when the Dijkstra's method is used for this purpose (see Methods).

In order to compare the performance of our method to that of the serial Isomap, we compared the residual variance in each sample molecule for a number of dimensions for each of the two embeddings. Residual variance is a measure of how different the generated embedding is with the original data. With increasing dimensionality, the number received gives an estimate of how much of the variance in the data is still unexplained. It is computed as the unit difference from the squared correlation coefficient at each dimension. The data matrix of N points is first subject to thorough analysis, by obtaining the shortest path tree that represents the connectivity of each of these points with every other point. The low-dimensional embedding obtained by eigen-decomposition of this tree forms the matrix with which the correlation coefficient is calculated. The residual variance is supposed to decrease with increasing dimensionality, as each dimension explains a fraction of the variance. A sphere when observed in two dimensions would be a circle. So, more dimensions are to be taken into account to reveal the true structure of any object. More the dimensions, less is the unaccounted information. For instance, for Oxytocin, the residual variance for the first dimension in Sequential Isomap (refer Table-2.2) is 0.448 which means 44.8 percent of the data is still unexplained, it decreases to 30 percent in the second dimension and it continues to decrease as more dimensions are added. The trend is reported for the first four dimensions for both serial and parallel versions. Similarly, in the Parallel Isomap (refer Table-2.3), the corresponding values for Oxytocin are 66.7 percent, 54 percent and they continue decreasing as more dimensions are observed. The difference in these values can be alluded to the way eigen-decomposition is performed in our version of the algorithm. A three dimensional projection of the embeddings generated by the serial and parallelized version of the algorithm is presented in Figure-2.1 in the same order as they appear in Table-2.1. The first three dimensions were embedded.

Another proof of quantitative validation comes with the least RMSD (Root Mean Square Deviation) computation for the two embeddings. This method is the same as described in the Data instance Reduction section. It is enlisted for the various protein molecules in Table-2.1 in the column named reconstruction error. Our implementation offers a good coverage even in lower dimensions.

The last column of Table-2.2 shows the performance of the serialized code in

Table 2.3: Parallel-Isomap trends in residual variance with increasing dimensions

Molecule	1	2	3	4	Time (min) for 8 cores	4 cores	2 cores
4did	0.054	0.032	0.00852	0.0056	92	132	156
Hgalanin	0.754	0.690	0.623	0.491	175	255	306
Pgalanin	0.421	0.334	0.259	0.062	152	243	290
Neuromedlin	0.974	0.783	0.642	0.341	125	165	198
Oxytocin	0.667	0.54	0.378	0.254	110	170	205
Vasopressin	0.834	0.657	0.278	0.109	162	190	230

Matlab. The last three columns of Table-2.3 do so for our version of the algorithm with scalability, depending on how much power of a CPU is harnessed. As is evident, the time consumption is much less for the parallelized code. The time consumption in the parallelized version depends on the number of cores of the CPU being employed for the computations. The results of Table-2.3 represent the time taken by a system with two quad-core processors, which means at a time eight threads are at work simultaneously for the parallelized portions of the code. The number of threads can be set for any program written in C using OpenMP [2]. The chunk of pseudo-code in Methods shows how to parallelize a simple for loop and how many threads one wishes to use can be set before executing the program, in the shell, using the command:

`OMP_NUM_THREADS=8`.

Since the number of cores here are eight, the performance isn't affected much by using more threads for this configuration of a machine. The average speed-up drops to about 1.5 when only four threads are at work and about 1.248 for two threads.

Besides the improvement in average time complexity due to betterment in computation of the shortest path tree, the algorithm offers better speed at every stage, when each function is compared in isolation. In future, we plan on attempting to do this even faster. The algorithm in its current form, only reduces the features of the data. Coming up with a way to reduce the noisy and redundant points themselves, is an area of ongoing research. As mentioned earlier, the Isomap algorithm is compute intensive. It executes in a way that hogs the system's resources, leaving all the other applications running on the system starved. In comparison, our parallelized version, efficiently makes use of the available memory, without having to write anything on the disk and hence doesn't hamper any other processes running on the system. The amount of data to be operated on is so large here, that programming platforms for vectorized data like Matlab and R sometimes cannot allocate memory for it. The coordinate data here is about 80 MB and the intermediate data files can be about 4GB. In situations like these, our version of the algorithm becomes the only resort.

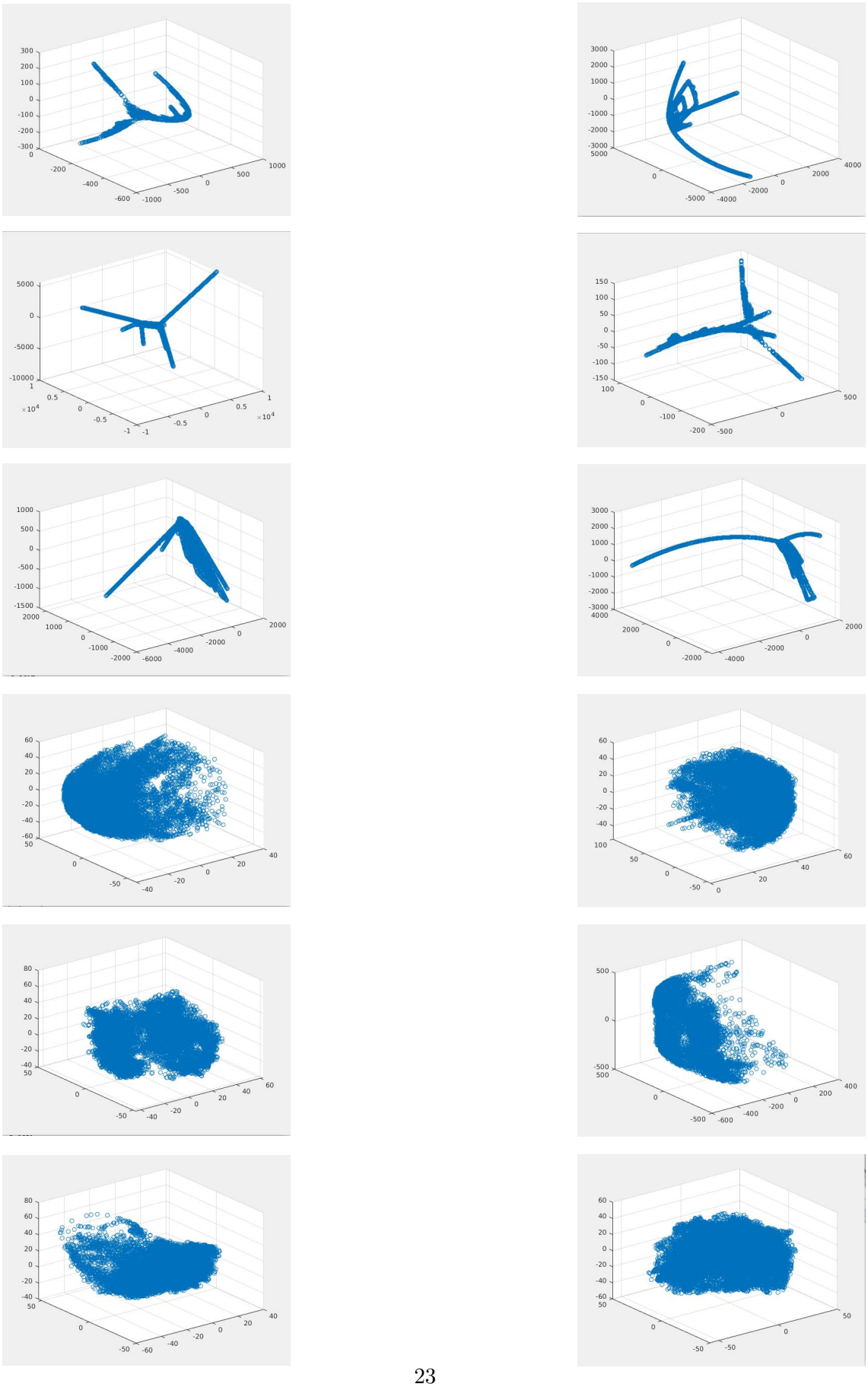


Figure 2.1: Isomap (left) and Parallel-isomap (right) Embeddings

Chapter 3

A NOVEL DATA INSTANCE REDUCTION TECHNIQUE

Representation of structurally significant data is indispensable to modern research. The need for dimensionality reduction finds its foray in varied genres viz-a-viz, Structural Bio-informatics, Machine Learning, Robotics, Artificial Intelligence, to name a few. The number of points required to effectively capture the essence of a structure is an intuitive decision. Feature reduction methods like Principal Component Analysis (PCA) have already been explored and proven to be an aid in classification and regression. In this work we present a novel approach that first performs PCA on a data set for reduction of features and then attempts to reduce the number of points itself to get rid of the points that have nothing or very little new to offer. The algorithm was tested on various kinds of data (points representing a spiral, protein coordinates, the Iris dataset prevalent in Machine Learning, face image) and the results agree with the quantitative tests applied. In each case, it turns out that a lot of data instances need not be stored to make any kind of decision. Matlab and R simulations were used to assess the structures with reduced data points. The time complexity of the algorithm is linear in the degrees of freedom of the data if the data is in a natural order.

3.1 The Problem

Dimensionality reduction is a process of expressing high-dimensional data using a low-dimensional representation, while trying to preserve the variance in the data as much as possible. Principal Component Analysis is now one of the age-old and well-established methods for feature reduction [26]. It takes as input a data matrix with M observations and N features. The algorithm performs an orthogonal linear transformation and produces the *principal components*, which are the eigen vectors of the covariance matrix of the original data that in decreasing order represent

the variance captured by them. The components are obtained from the Singular Value Decomposition of the data matrix. More details of the core algorithm can be found in [27, 28].
There are many approaches to obtaining these principal components, some of which are categorized as *robust*. Such algorithms, take into consideration the sparseness and possible corruption in the values of the data matrix [29]. We use one such approach called the *spherical* PCA. The data points in this approach are projected onto a sphere. The sphere is centered around the biggest cluster in the dataset and the radius is so chosen that most data points are covered. It offers a clever way of dealing with the outliers inherently present in the data. A single outlier affects the average of the data tremendously and ultimately the direction of principal components [30].
Real world structurally significant data is often so huge that its processing and analysis is long and tedious. Such datasets are also intensive on the system's memory. Also, in higher dimensions, the data becomes sparse. Consider the ratio of the areas of a square of length r and that of a circle of radius r, which is about 0.3183. Now consider the ratio of the volume of a cube and the volume of a sphere in three dimensions, the ratio now becomes approximately 0.2387. This ratio continues to decrease as the dimensions increase, which indicates that in high dimensional data the significant information moves towards the boundaries. The notion is referred to as the *curse of dimensionality*. Therefore, there arises a need for an algorithm that would make a decision as to whether a particular data instance in high dimensionality contributes to the shape of a structure or not. In particular, the motivation for this work was forked off because of dealing with humongous molecular data sets[19, 20]. These data sets are often in the form of text files of the order of 25 MB. Each row of these files represents a conformation that the molecule can have. Each conformation is represented by a set of attributes. These attributes can be normalized and reduced using various feature reduction techniques like PCA itself. Studies have shown that not all of these conformations need to be worked with to understand macro-molecular dynamics [7, 8, 9, 10]. Our algorithm attempts to reduce these conformations.

The goal here is to obtain a reliable method to reduce the number of observations for data sets while losing as little information as possible. It finds relevance in the fact that lesser points would need less storage space and also reduce computing times of algorithms used to analyze these data sets. PCA processing in this work is used so as to obtain three distinctive features of the data that among themselves preserve the most variance. The algorithm then assesses the information content offered by each point. The assessment is done by the relative positioning of the points in the space defined by these first three principal components. As the

Abstract claims, the algorithm has been shown to produce noteworthy results in eclectic data sets. In image processing, the classical techniques of edge detection and image recognition are of paramount importance. The algorithm can be an aid for all of these procedures. Details of more widespread applications of image processing with efficient data reduction and its importance in the medical world can be found in [31] and [32].

3.1.1 Literature Survey

Existing algorithms for data instance reduction are broadly divided into, incremental, decremental, batch and mixed [33, 34, 35, 36, 37]. The incremental algorithms begin with a null set and data instances are added to it depending on the result of the algorithm. The decremental algorithms, on the contrary, begin with the entire set of instances and depending on the decision offered by the selection algorithms, instances are taken out from the set one had at the beginning. The batch algorithms function in a way that each instance is first analyzed and then a decision is made as to which ones to keep. Mixed algorithms begin with a preselected set of instances and the process then continues to figure whether instances should be deleted or added. The proposed algorithm falls into the decremental bracket.

An evaluation of the age-old techniques of instance reduction is explained in [33]. Another work that performs an elaborate evaluation of existing algorithms and presents two ways based on Locality Sensitive Hashing [35] is presented in [34]. Another novel approach based on similarity calculation between instances and then clustering is presented in [36]. Another novel entropy based approach is presented in [37]. All of these algorithms are aimed to produce the best training set to produce accurate classification of a dataset. The algorithm proposed in this work explores a range of datasets and has been shown to work in each scenario. Comparative results and their analyses are provided in section 3.3.3.

3.2 The Algorithm

As mentioned earlier, the first step is to perform spherical PCA on input data matrix. The data matrix could be a distance metric for all the data instances or a combination of attributes that measure the similarity of each data instance. If the data has non-numerical information, then that information needs to be translated into numerical data. Computing spherical principal components is a well known algorithm and we followed the procedure described in [29, 30, 38]. We wrote a short Matlab script for the purpose, it takes as input the data matrix (with only numeric features) and the number of dimensions (principal components) desired of all the data instances. The output is a matrix file that retains all the points

with the number of desired principal components. We work with two and three dimensions, in order to better visualize structurally significant data. It is observed that over eighty percent of the variance is explained by the first three principal components in all kinds of data the algorithm was tested on, although it is not always the case. The method can be readily extended to higher dimensions. PCA redoes the features of the data set and produces a new set in decreasing order of variance captured by each of them. The relative values of these high variance capturing principal components is used to pick the relevant data instances.

3.2.1 General Outline

The data set at the beginning has N rows and M columns. The PCA processed data with N rows and three columns (first three principal components is fed to the algorithm). For a blind dataset (or a data set of known random nature), a sort should be performed based on the first principal component so that the points can be processed in order. Following is the pseudo-code that describes our approach:

```
1. //2-D case: data set has N points
2. for i=2:N-2 {
3. a = ratio of slope between i-1, i and i,i+1
4. if(a < threshold)
5. remove point i
6. }
7. //3-D case: the data set now has Q points
8. for q=2:Q-4 {
9.  P1 = position vector of point q-1
10. P2 = position vector of point q
11. P3 = position vector of point q+1
12. P4 = position vector of point q+2
13. N1 = cross product(P1-P2, P1-P3)
14. b = angle between N1 and P4
15. if(b<pi/5 || b>3*pi/5)
16. remove point q+1
17. }
```

3.2.1.1 Projection Score

We seek an optimum value of the slope threshold (in line-4 of the algorithm above) and the angle between planes (line 14). Their optimality is decided by the what is called the *projection score* of a dataset. It implies how much a given point contributed to the variance of the entire data set. A mathematical formulation of a proof of correctness has been adapted from a method described in [39]. This work

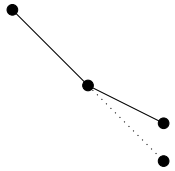


Figure 3.1: *Elucidation of the Algorithm. The middle point would be kept or rejected depending on where it lies with respect to the other two.*

formulates a way to credit the informativeness of a variable in huge data sets. We use this formulation on data instances instead of features. For this, first the matrix is multiplied by its transpose to obtain an empirical covariance matrix. Let A be the data matrix of dimensions $N \times M$. The covariance matrix of dimension $N \times N$ is:

$$Cov(A) = A * A^T$$

Next, the eigen-values of the covariance matrix are computed. . Let λ_x denote the eigen-value in x^{th} dimension. The ratio of the sum of these values in rejected data points to the total number of points is referred to as the projection score of these data instances. Let S denote the set of points eliminated by the algorithm, projection score α can be denoted by the following equation.

$$\alpha = \frac{\sum_{x \in S} \lambda_x}{\sum_{x=1}^{N} \lambda_x}$$

The lower this number is, the lower is the information content offered by this set of points. This number is found to be of the order of 10^{-3} at the very least in almost all data sets. The Projection Score column of Table-3.1 shows these results.

3.2.2 2D case

To decide whether a point can be removed from a projection, we measure how much information it adds to it. Intuitively, if three close by points are co-linear, the middle point does not add new information to the projection. To obtain information about these points, the first two principal components are traversed in the order in which they exist in the data file and the information content offered by each point is assessed as follows: Given three points p_1, p_2, p_3 at coordinates $(x_1, y_1), (x_2, y_2)$ and (x_3, y_3), respectively, then the threshold is defined as the ratio of the slopes of the lines $p_1 - p_2$ and $p_2 - p_3$:

$$\frac{\frac{y_2 - y_1}{x_2 - x_1}}{\frac{y_3 - y_2}{x_3 - x_2}} = \frac{y_2 - y_1}{x_2 - x_1} \times \frac{x_3 - x_2}{y_3 - y_2}$$

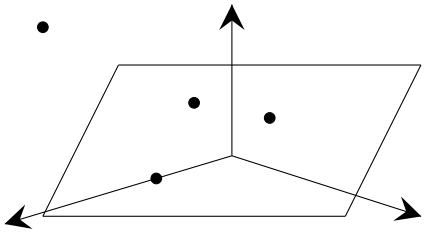


Figure 3.2: *The Algorithm in three dimensions. Amongst the four co-planar points the one encountered third would be rejected or kept depending on how much the plane is to be tilted in order to include the point that is out of the plane.*

If this change of slope is above a threshold, the middle point is kept, otherwise it is discarded. Figure-3.1 illustrates an example of three points. The middle point is the one being investigated. The method checks for collinearity. The ratio of slopes that constitutes collinearity depends on the kind of data. To find an optimal value we conducted a binary search starting from a given threshold, as detailed below. The threshold values for the various data sets are reported in the last column of Table 3.1. This value of the threshold maintains much of the variance in the data, while rejecting the most number of points. The process is explained in details in Section 3.2.4. Decreasing this number any further may eliminate more points from the set but loses too much of its variance, according to the score described below.

3.2.3 3D case

A straightforward extension of this algorithm can be applied to three dimensions, where we seek to eliminate points that do not contribute to the projection by testing whether they are *co-planar* with three other close-by points. Since every three points define a plane, a fourth point on the plane does not contribute to the projection. The algorithm is performed on the reduced data set received from the two dimensional projection. As before, the points again are traversed in the order they appear in the dataset (or the sorted order based off of first principal component if the data was know to be random). In a sequence of four points, the threshold (the definition of threshold here is the same as in 2-D case) of angle required to tilt the plane on which the first three points lie helps in deciding whether the fourth point in this sequence is informative or not. If this change of angle is above a threshold, the fourth point is retained, otherwise it is rejected. The details of how this is achieved are in the next section. Figure 3.2 provides for visualization of this procedure in three dimensions.

3.2.4 Determine the Threshold

As mentioned earlier, the *thresholding* here is paramount and depends on the kind of dataset being dealt with. For a new (blind) dataset, we start with an initial

slope ratio value (for the 2-D case). Starting with 0.5 we assess the results based on the percentage of points eliminated and the projection score defined below. Assessing the results, there are three options, this threshold is either adequate, too high, or too low:

1. If the initial value 0.5 is too high, the reduced dataset will be too small. Per our definition, this means over 75% of the data is lost and the projection score is 10^{-2} or higher. A combination of the percentage of eliminated points and the projection score decide whether the subset obtained is acceptable or not. For example, in the Swiss Roll dataset, described in the following section, the percentage of points eliminated is at slope threshold of 0.5 was over 75%, but the projection score is much lower (3.1, column 1) which meant that a slope threshold of 0.5 is not too hight this dataset. But for the datasets that do fall in this category, the next step would be to perform a binary search between 0 and 0.5, to obtain an optimal threshold. The next value of threshold to try would be 0.25, if the projection score is still high, try 0.125 and so on. The maximum number of trials in this study, 15 iterations, were needed for the face image dataset.

2. If the initial value 0.5 is too low, the dataset ends up retaining most of its points. By our standards, it happens if only 15% or fewer points of the original data are eliminated. In this case we perform a binary search between 0.5 and 1 until the convergence criterion mentioned above in the first case is achieved. All of the protein datasets in this study fall into this category, where (even though we began with a much lower threshold for experimentation purposes), the eliminated points were well above 60% and the projection scores as low as 10^{-5} were achieved.

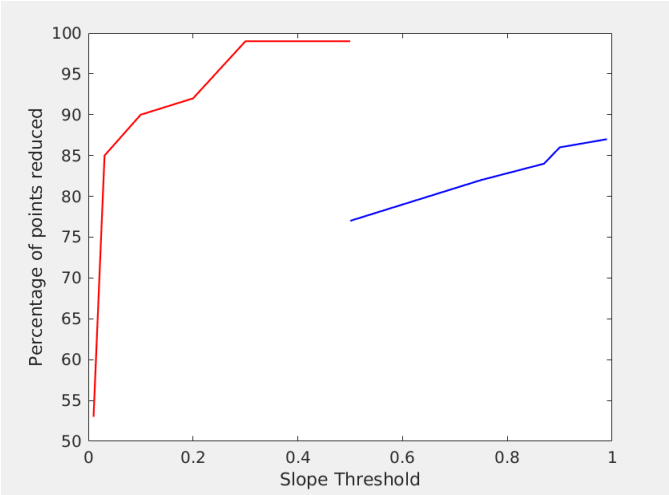
3. The slope threshold of 0.5 is considered near adequate if the percentage of points eliminated are between 15 and 75. In this case, the threshold and the corresponding projection score should be considered for the values of slope threshold of 0.4 and 0.6.

    a) If the percentage of points decreased is less than 50 and the threshold of 0.6 results in an even smaller dataset while maintaining (or lowering) the projection score, we perform a binary search between 0.6 and 1. If not, the search should be between 0.5 and 0.6. As soon as the projection score becomes higher, stop and return the slope threshold for this dataset.

    b) If the percentage of points eliminated is over 50, we still try a higher slope threshold of 0.6 because the goal here is to be able to represent data with as fewer points as possible while capturing maximum variance. The spike in projection score is evident of the fact that informative point(s) have

been removed from the dataset. Reducing the threshold to 0.4 results in a comparatively larger dataset while maintaining the projection score, we conduct a binary search between 0.4 and 0.5. If the projection score at 0.4 becomes lower, it is indicative of the fact that at the threshold of 0.5 there has been a loss of few informative points. The search should then be between 0 and 0.4. As soon as the projection score becomes higher, stop and return the slope threshold for this dataset.

5-6 trials are usually enough for the purpose. Most machine learning datasets fall in this category.
Figure-3.3 shows the process for the two extremes. In both the cases the stopping point for the threshold for the process is decided by the projection score at that point for the corresponding dataset.



;

Figure 3.3: *Slope Threshold versus the percentage of points reduced. In both the cases the stopping point for the threshold for the process is decided by the projection score at that point for the corresponding dataset.*

4. As mentioned earlier, the next step is to try to eliminate even more points from the data set by progressing onto the 3-D version of the algorithm. The decision of rejecting a point, in this case, is determined by how much the plane created by the previous three points is to be tilted in order to accommodate the current point. The process of determining this *angle threshold* is the same as for the slope threshold. The angle that we begin with is $\pi/2$ (and $3 \times \pi/2$ to account for the negativeness of the angle). Depending on the size of the reduced data set and the projection score, a binary search can be started between π and $\pi/2$ or $\pi/2$ and 0.

Also, besides the projection score and the percentage of points eliminated, each dataset has its own method of evaluation that presents insight. The goal of *Su-*

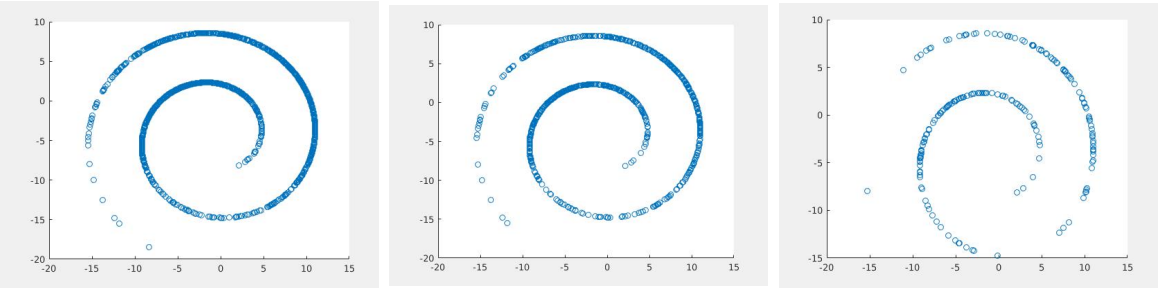


Figure 3.4: *2-D PCA projection of the Swiss Roll dataset with 1600 (left), 833 (middle) and 210 points (right).*

Table 3.1: Projection Score trends in various data sets

Data Set	Projection Score	% of points eliminated	Reconstruction Error	Slope Threshold
Swiss Roll	$3.204 * 10^{-4}$	86.875	0.006	0.99
Human Galanin	$2.159 * 10^{-5}$	68.954	1.562	0.76
CDC42	$1.42 * 10^{-3}$	69.875	1.022	0.78
Vasopressin	$1.893 * 10^{-5}$	74.302	0.358	0.82
Lena Image	$2.03 * 10^{-5}$	53.36	0.13	0.00001
Iris Dataset	$7.37 * 10^{-9}$	52.67	0.02	0.16
Isolet Dataset	$6.4 * 10^{-3}$	40.23	0.14	0.46

pervised Learning is to exploit the information known about the dataset. The datasets that represent a shape, like the Swiss Roll and Face Images, can be evaluated on the basis of their simulation as well. In the context of machine learning, the reduced dataset is treated as a *training set* to classify the rest of the data, details can be found in section 3.3.2.1.

3.3 Results and Discussion

3.3.1 Simulation Results

3.3.1.1 Swiss Roll Dataset

Figure-3.4 (left) shows the PCA projection of a data set representing a Swiss Roll [18, 21]. It is a two dimensional projection that uses 1,600 points to represent the structure. After applying just the 2-D version of the algorithm, about half of these points were found to be redundant. Figure-3.4 (right) shows the plot of the PCA projection of the remaining points. Only 833 points were used to construct this plot. As mentioned earlier, this is one of the datasets that represent a shape. Apart from the projection score and the percentage of elimination of data instances, the simulation results are quite instrumental in determining the validity of a reduced subset of points. In search of validation, a very different kind of structurally significant data set was chosen next.

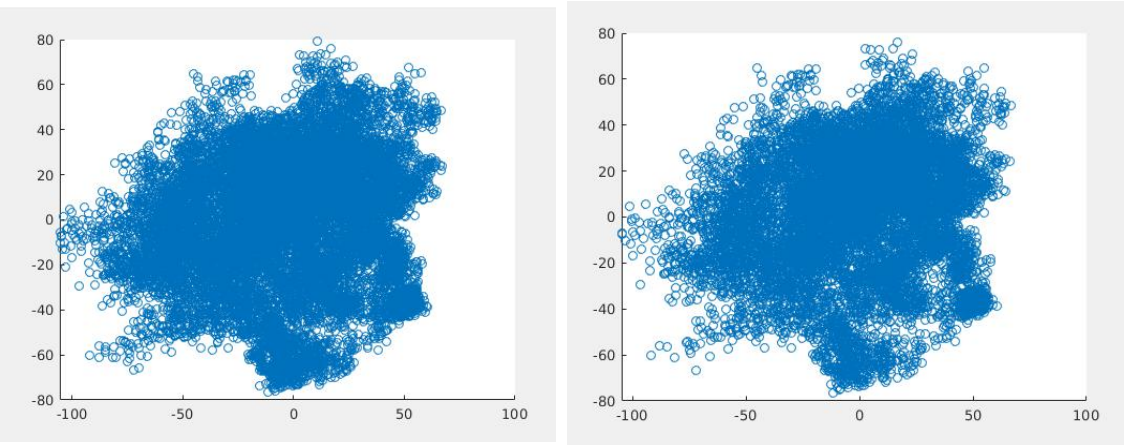


Figure 3.5: *2-D PCA projection of the Human Galanin dataset with 22750 (left) and 15680 (right) points respectively.*

3.3.1.2 Protein Datasets

We used simulations of protein structures. As mentioned in section 3.1, these molecular datasets are in the form of matrices, each row of which represents a conformation that the protein can attain in its trajectory. The data was created using Molecular Dynamics (MD) simulations [40]. If a molecule is composed of N atoms, it has $N \times 3$ attributes (columns), taking into account three-dimensional coordinates of each atom, that distinguish a conformation. The two structural extremes of the molecule are represented often with tens of thousands of conformations, making the data humongous. Human Galanin is one such example. It is a neuro-peptide and is a known heavy protein molecule with ample structural niceties. Figure-3.5 (left) shows a two dimensional plot of the PCA projection of this macro-molecule with 22750 data points. The algorithm does away with about two-thirds of these points. Figure-3.5 (right) is a PCA plot of the reduced data and was constructed with 15,680 points.

These results were obtained with just the application of the two-dimensional version of the algorithm. To do further tests, a set of medium to large protein molecules with well-established structures were chosen. Figure-3.6 (left) shows the three dimensional plot of the PCA projection of one such protein, Cdc42, with 20,000 instances. Cdc42 is a protein from the Ras superfamily, involved in regulation of the cell cycle and has been shown to be involved in cancerous processes [41]. When subjected to the two dimensional version of the algorithm, a data set of about 15,000 points was obtained. The number of points obtained subsequently by progressing to a plane was 4,985. Figure-3.6 (right) shows a three-dimensional plot of these points. When the Swiss roll data set is again subjected to the progression of the algorithm, a set of 210 points was obtained. A two-

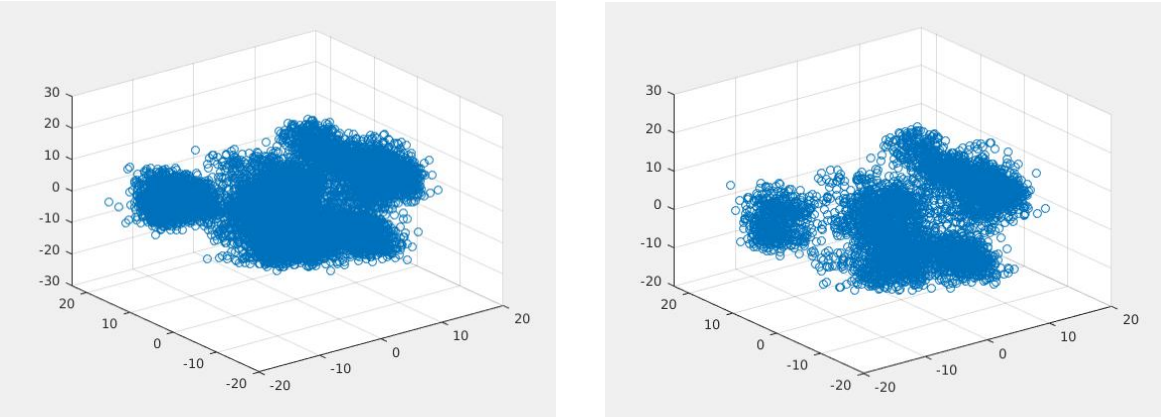


Figure 3.6: *3-D PCA projection of the CDC42 dataset with 20000 (left) and 4985 (right) points respectively.*

dimensional plot of these points is shown in Figure-3.4 (right). Figures-3.7- left and right show the same for Vasopressin, a hormone.

3.3.1.3 Face Image

In light of the fact that the algorithm preserves a great deal of variance in structurally significant data, testing it on facial images seemed like a test that would consolidate its function. In image processing, efficient algorithms for de-noising an image are imperative. Many filtering processes exist for the purpose [42], [43]. The Lena image, shown in Figure-3.8 was used as a test dataset here too. This image is pervasive and was procured form the Internet with a basic Google search. Its jpg file format was then converted into coordinate data using the method described in [44]. A two-dimensional projection of these points is shown in Figure-3.9 (left). Once a coordinate matrix is obtained, it is PCA processed and the algorithm is performed as with other data sets. The reduced data set obtained here contained less than half the points in the original image and its plot is shown in Figure-3.9 (right). In this data set, the algorithm discarded almost all the points (only three points were retained) for the values as low as 0.01. In this case the slope was decreased gradually until a considerable number of points were retained, which when plotted recreated the structure of the image. This number was 0.000001. Among the datasets used, this was the most complicated one, structurally. It had more nooks and crannies to cover in order to capture the variance and hence the slope threshold here is the lowest.

3.3.2 Quantitative Analysis

An advantage of using PCA processing presented itself in search of validation proofs. PCA is linear and the original data matrix can always be recreated. This

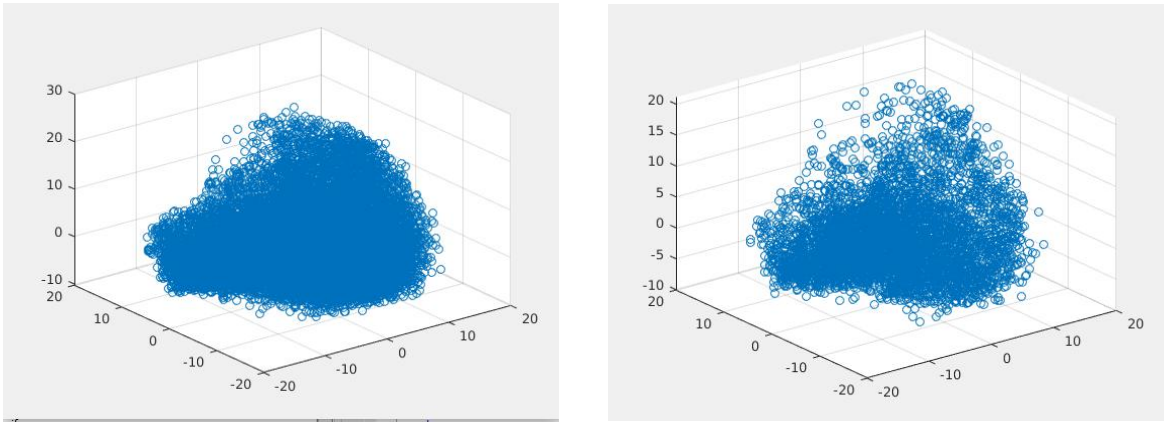


Figure 3.7: *3-D PCA projection of the Vasopressin dataset with 25000 (left) and 5320 (right) points respectively.*

Figure 3.8: The original Lena image.

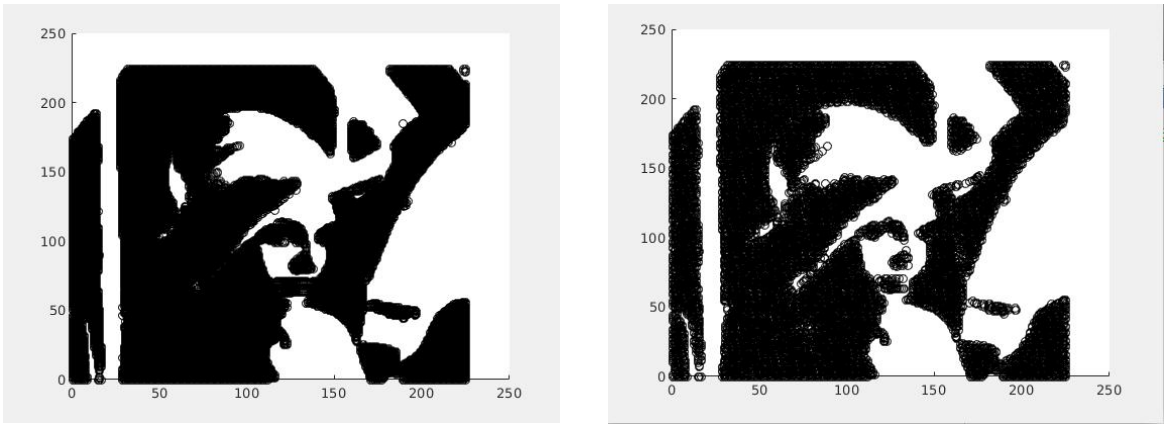


Figure 3.9: *2-D projection of Lena data set with 28,080 (left) and 13,097 (right) points respectively.*

fact was exploited to formulate a number termed the reconstruction error. First, the points eliminated by the algorithm are removed from the original data, this leaves the data matrix with as many points as the PCA projection returned by the algorithm (but all the original features/attributes). PCA projection of this smaller data set is then obtained. Then, the root mean square deviation (RMSD) of these two embeddings is calculated. This method first eliminates the translation component by shifting their center of mass to the same place, and then it finds the optimal rotation between the two sets using Singular Value Decomposition. The difference between the two structures is reported in the form of an error. Let the two matrices being compared be, A and B with dimensions $N \times M$. First, the centroid of the two is found, both the molecules are dragged to the origin by subtracting from each point the value of the centroid. The RMSD between the structures can then be calculated as under:

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(\sum_{j=1}^{M}(a_{ij}-b_{ij})^2)}$$

Here, a_{ij} and b_{ij} are the corresponding elements of A and B respectively. The number returned by equation above indicates how similar the two structures are. It is enlisted for the various data sets in Table 3.1 in the column named reconstruction error. Also, as mentioned earlier, in the data sets the algorithm was tested on, the first three principal components preserve over eighty percent of variance inherent in data. The PCA projection on the reduced data set follows these trends very closely. For example, in Cdc42, the first three principal components capture respectively 42.082, 34.35 and 23.569 percent of variance. In the reduced form of Cdc42, the first three principal components capture 49.436, 28.34 and 22.22 percent of residual variance respectively. This trend is observed in all of the other data sets too. It is reported in Table 3.2 for the original data sets and in Table 3.3 for the reduced form of these data sets. If the corresponding principal components capture a similar amount of variance, this indicates that the algorithm retains those points that actually contribute to generating the variance values in the original data.

3.3.2.1 Machine Learning Datasets

The slope threshold defined in Methods and enlisted for all data sets in Table-3.1 plays a crucial role in the analysis of datasets for classification and regression. These datasets had non-numerical attributes and so were preprocessed to either convert them to numbers or just get rid of them depending on what they represented. Subsequently, the rest of the algorithm was applied similarly as with other datasets. These were the only datasets that didn't represent a structure and unlike the protein datasets, a better understanding of the structure is not what

[1]Principal Component

Table 3.2: Residual Variance trends in original data sets

Data Set	Variance in 1st PC[1]	Variance in 2nd PC	Variance in 3rd PC
Swiss Roll	55.55	27.87	16.59
Human Galanin	41.22	34.05	24.72
CDC42	42.082	34.35	23.569
Vasopressin	47.75	33.23	19.012
Lena Image	52.88	35.11	13.02
Iris Dataset	78.37	13.59	8.04
Isolet Dataset	57.13	31.06	11.82

Table 3.3: Residual Variance trends in reduced data sets

Data Set	Variance in 1st PC	Variance in 2nd PC	Variance in 3rd PC
Swiss Roll	63.44	27.30	9.26
Human Galanin	43.112	29.43	27.46
CDC42	49.436	28.34	22.22
Vasopressin	53.68	28.98	17.33
Lena Image	56.07	32.23	10.63
Iris dataset	86.65	10.35	3.00
Isolet Dataset	58.59	31.01	10.40

was sought here. Nevertheless, like the protein datasets, a pictorial representation makes sense in terms of clusters of similar or close by points [45], after a projection of the original data is obtained (PCA, or any other feature reduction method for that matter). The first dataset used here was the Fisher's Iris dataset [46, 47]. It is a collection of three species of Iris flowers. It has 50 instances each of Iris Setosa, Iris Versicolour and Iris Virginica. Iris Setosa is the one that is linearly separable from the latter two. Figure-3.10 (left) shows the PCA projection of the entire dataset. The instances differ on the basis of four attributes, namely, sepal length, sepal width, petal length and petal width, all in centimeters. Any value of the slope threshold here, as expected, gives two disjoint sets of data instances. Table-3.1 reports this value for the Iris data set to be 0.16. The reduced data set so obtained is the smallest such set which when used as a *training set*, produces a 100 percent correct classification of Iris Setosa for the rest of the data set, which here is used as the *validation set*. Figure-3.10 (right) shows the PCA projection of the reduced Iris data set. Perturbing the slope threshold further to include more data samples, as expected gives the same results. Increasing the slope threshold any further to obtain an even smaller training set mis-classifies a few data instances. A slope threshold of 0.17 produced a mis-classification of 1.92 percent. R simulations with five-fold cross validation using a Support Vector Machine with a linear kernel were used for verification purposes [48]. The reconstruction error in Table-3.1 and the trends in residual variance of Tables- 3.2 and 3.3 bear the same explanation as for the other datasets. In order to assess the scalability of the algorithm, another multivariate dataset was chosen to test the abilities of extraction of a suitable

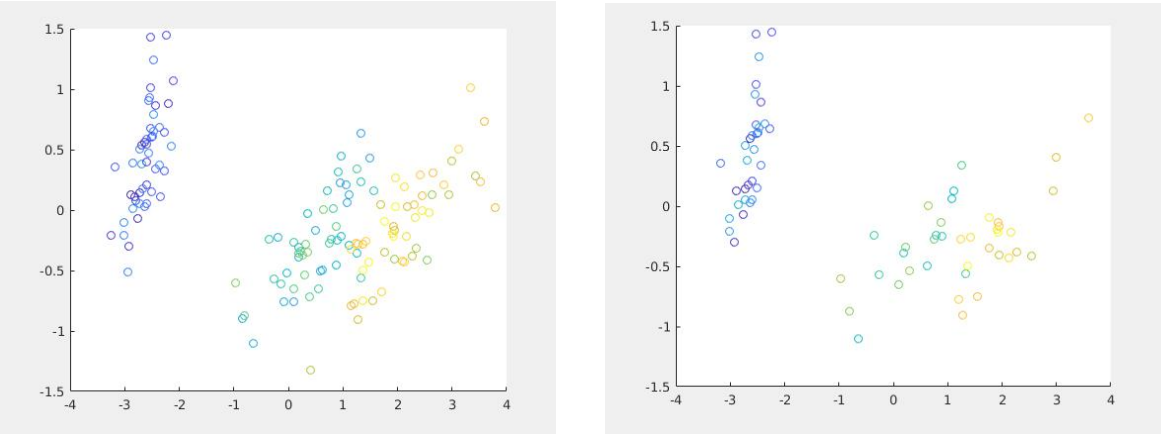


Figure 3.10: *Full and reduced Iris data set with 150 (left) and 71 (right) data instances respectively.*

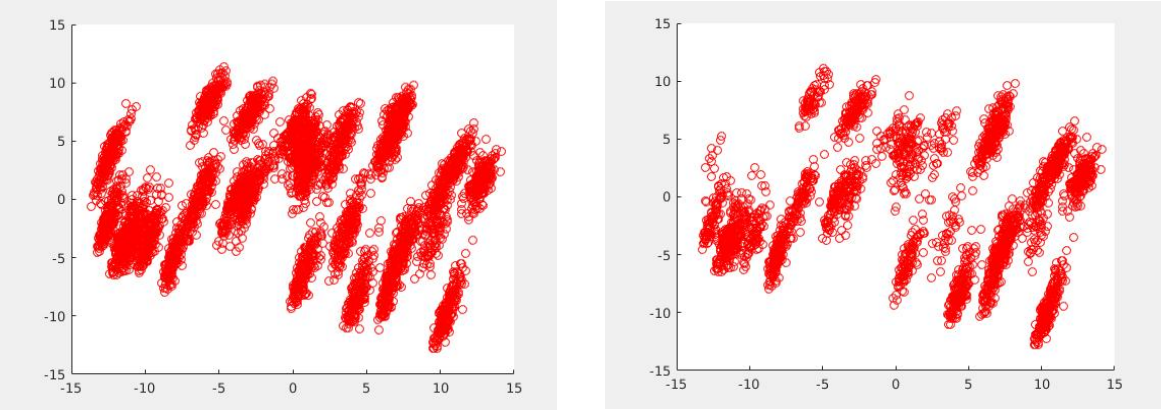


Figure 3.11: *Full and reduced Isolet data set with 6238 (left) and 3164 (right) data instances respectively.*

training set that produces better classification. The Isolet dataset was generated for speech recognition [49, 50, 51]. It contains 52 samples of voice for 150 subjects. Each person uttered a letter of the English alphabet twice. The dataset, as on the Machine Learning group's website of University of California, Irvine,[**?**] is divided into five groups of 30. Four of these are used as the training dataset and the fifth one as the validation set. The attributes of this dataset are described in the paper [49]. They include spectral coefficients like contour features, sonorant features etc, all real numbers describing a sample of a voice of a subject. The goal of this work was to produce a 26 way classification, to identify which alphabet was spoken by a subject. It achieves over 95% accuracy. When this dataset is subjected to the algorithm described in Methods, it again produces a reduced dataset eliminating 40.23% of the instances. The PCA projections of the full and reduced forms are shown in Figures-3.11 left and right respectively. Evident from the projections, the reduced dataset clearly produces the same clusters. This dataset when used as the training set also produces the same classification. The procedure was the same as described for the Iris dataset. The Isolet dataset has more groups to classify than the Iris and many of these groups (each one representing an alphabet of the English language) are phonetically more similar than others, so the percentage of points eliminated here should be lesser than the Iris dataset because the data here is more diverse. This was found to be true, and the algorithm distinguishes between these two very similar, real-valued, multivariate datasets by a margin of 12.44% in context of data instance elimination.

3.3.3 Comparison with Other Instance Reduction Methods

As pointed out in section 3.1.1, all of the other data instance reduction methods have only analyzed the machine learning datasets. Therefore, their method of validation is using the reduced dataset to produce a classification and measure its accuracy. The average accuracy of classification and the percentage of points eliminated for these methods versus the proposed algorithm is in Table-3.4. The datasets used here are the same as on UC Irvine's website, including the Iris and Isolet datasets described in the previous section. The values for the proposed algorithm in the table have been evaluated to produce the most accurate classification. If saving space is the motivation, the slope threshold can be compromised with the accuracy to produce even smaller datasets.

With the exception of LSH, all of the algorithms in Table-3.4 are linearithmic (NlogN) at best. LSH, like the proposed algorithm (provided a sort on the data is not needed), is linear, but works in a very different way. It depends on formulation of hash functions over the dataset that are sensitive to data instances. In other words, a family of hash functions represents the various categories a data instance

Table 3.4: Comparison of the proposed algorithm with other instance reduction algorithms.

Algorithm	Average Accuracy for classification	% of points reduced
Instance Based Learning Algorithms [33]	78.85	16.13
DROP-3 [33]	81.14	14.31
Entropy based algorithm[37]	≈ 85	88.72
LSH based algorithm [34]	≈ 90	≈ 60
Instance reduction Algorithm [36]	78.23	≈ 50
Proposed Algorithm	≥ 90	≈ 70

can belong to. Each data point's compatibility with each of these functions is evaluated. One point representative of each category is chosen to be a part of the reduced data set. The point chosen is the first one that makes its way into a certain category. The proposed algorithm works better for the purpose as it is adherent to the dataset. It does a better job in evaluating the information content offered by a particular data point. Only one point per category is not always enough to represent a category in its entirety. How many and which of the points of a certain category should be chosen is the strength of the proposed algorithm. Also, formulation of a class of functions is a task in itself and not having to do so, betters the time complexity. The proposed algorithm also performs better when the data is highly sparse, or in other words the categories for classification are of the same order as the number of data instances available.

Chapter 4

SAMPLING OF INTERMEDIATE PROTEIN CONFORMATIONS

Datasets representing complex protein structures are high dimensional and hence tedious to work with. Efficient and effective dimensionality reduction of these datasets is therefore paramount. Having fewer data samples with only the attributes that capture the most variance of the data, makes for quicker and precise analysis of these structures. In this work we make use of two dimensionality reduction methods- one for reducing the number of instances and another for feature reduction. The refined dataset so obtained is then subjected to topological and quantitative analysis. In this step we perform *hierarchical clustering* to obtain different sets of conformations. The structures represented by these conformations are then analyzed by studying their high dimension *Betti numbers* to establish or disparage their distinctiveness.

Isolating transient protein conformations is difficult to do experimentally due to the fleeting nature of these structures. Doing so is imperative for understanding and characterizing protein function and dynamics. This work presents a way to do the same, using mathematical tools for extracting meaningful information inherent in these structures.

4.1 Problem Statement

The protein folding problem, aims at predicting the correct protein structure from its sequence, it is an important problem in Biology. The conformational search problem is a related problem that aims at characterizing the conformational space of proteins in order to find minimum energy regions corresponding to highly populated structures and characterize dynamic events such as folding or docking [52, 53]. The potential energy landscape of a protein is immense. A typical protein molecule has hundreds of amino acids and several thousand atoms. Consequently, the num-

ber of configurations that a molecule can attain are humongous and are practically impossible to enumerate. This has given rise to a foray of methods that attempt to model the actual pathways that a protein undertakes while transitioning from one conformation to another that ultimately help in pinpointing the conformations that actually get attained in the entire process. In this work we present a combination of efficient ways to first reduce the dimensionality of the diverse datasets that represent the molecule and then use a number of filtering techniques to identify clusters of imminent conformations. The molecules used range from small (Oxytocin and Vasopressin) to medium (CDC42) to large (GroEl). Details about them can be found in the Results section. The main contributions of the paper can be summed up as under:

1. Data Refinement- Using previous works described in sections [54] and [55], we come up with a space efficient way to represent molecular data. The datasets in use have just enough instances and attributes that capture maximum variance.

2. Use versions of hierarchical clustering, depending on the molecule, to sample different conformations of a molecule. Details can be found in section 4.2.2.1.

3. Use advanced algebraic topology to analyze distinctiveness among various clusters of conformations.

4.1.1 Generation of Data

The molecules chosen with in this work are diverse and very different in their function and dynamics. There are two distinct ways in which the various conformations and their periphery was recorded. Although the datasets that resulted represented the same information. For, oxytocin, vasopressin, hGalanin, pGalanin and CDC42 Molecular Dynamics(MD) simulations were ran. Backbone resolutions were generated for all of them. The initial linear peptide was modeled using the Chimera software [56] as an idealized helix. Hydrogens and solvent were added using the AMBER software package [5]. Simulations were performed in constant volume (NVT) in an orthorhombic box using the TIP3 solvent model [57] to simulate infinite dilution. Periodic boundary conditions were applied using the nearest image convention. The overall charge of the system was kept neutral for the use of particle mesh Ewald summation to calculate electrostatic charges [58]. The simulations were carried out with the NAMD package [59] using the AMBER ff03 force field [60]. The other method used for the heavy GroEl molecule is a Monte Carlo (MC) [6] based conformational pathway search. The search begins with the PDB (Protein Data Bank) format of one conformational extreme and expands following a biased Rapidly-expanding Random Tree (RRT) algorithm to simulate the pathway that can be undertaken to reach the goal conformation [61]. At every

iteration a parent protein conformation is chosen from pool of new conformations, the one selected is the one which has its energy below a threshold [1]. The new conformation is added to the pool[2] if its RMSD (Root Mean Square Deviation) is closer to the goal.

4.1.2 Literature Review

Algebraic topology approaches are hugely unexplored in the context of sampling protein structures. The work in [62] describes the use of topological signatures, which the authors call evolutionary homology (EH) barcodes, reveal the topology-function relationship of the network and thus give rise to the quantitative analysis of nodal properties. The proposed EH is applied to protein residue networks for protein thermal fluctuation analysis, rendering accurate B-factor prediction for a number of proteins. A thorough review of the emerging applications of topological methods to genomics is presented in [63]. An insightful piece presenting concisely the ramifications of these approaches are elucidated in [64].

4.2 Procedure

4.2.1 Dimension Reduction

Both the feature and the instance reduction methods used here have been discussed at length in Chapter 2 and 3 respectively. The combined result of these processes is the input here.

4.2.2 Topological Analysis

After obtaining a reduced embedding that represents the dataset, the next step is to slot the data to sample different conformations inherent in these embeddings. To ascertain the number of conformations that can be extracted, we use two methods, namely, hierarchical clustering and persistent homology. Persistent Homology is also used as the next step but here both these methods reveal the same structural niceties in each dataset and hence act as a proof of validation for the process.

4.2.2.1 Hierarchical Clustering

Hierarchical Clustering is a method for identifying similar groups in a dataset. Built-in function in R, *hclust* is used for the purpose. Hierarchical Clustering is an alternative to k-means clustering wherein, one has to pre-declare the number of clusters sought, there is no such stipulation with hierarchical clustering. The

[1] it is predefined, takes care of whether a conformation would be feasible, keeping in mind atom collisions

[2] the conformational pathway

results of hierarchical clustering can always be presented coherently in the form of a dendrogram. Depending on how similar or dissimilar the data instances are, it can be divided into two categories:

Agglomerative Hierarchical Clustering It works in a bottom-up fashion. Each data instance is considered a single element cluster to begin with. At each step of the algorithm, two most similar clusters are combined into one. The process continues until all instances have been combined into one big cluster containing all the data instances.

Divisive Hierarchical Clustering This method works complementary to the previous one. Here, all the data instances are considered a point of one big cluster at the beginning. At each iteration of the process, the most heterogeneous cluster is divided into two. The process continues until all the data instances are a single element cluster.

Agglomerative form of hierarchical clustering is good for more heterogeneous data that has a large number of small clusters to identify. On the other habd, divisive clustering expectedly, is good at isolating big clusters. We use both these approaches depending on which molecule we are dealing with. The molecular datasets that are known to undergo large scale conformational changes like Calmodulin, Adenylate kinase and GroEl were subjected to agglomerative hierarchical clustering and the others to the divisive one.

4.2.2.2 Persistent Homology

Persistent homology is a method for computing topological features of a space at different spatial resolutions [65]. To find the persistent homology of a space, the space must first be represented as a *simplicial complex*. A distance function on the underlying space corresponds to a *filtration* of the simplicial complex, that is a nested sequence of increasing subsets. A brief review of these terms is as under [66]:

Simplicial Complexes An abstract simplicial complex is given by the following:

- A set Z of vertices or 0-simplices.
- For each $k \geq 1$, a set of k-simplices $\sigma = [z_0 z_1 ... z_k]$, where $z_i \in Z$.
- Each k-simplex has k + 1 faces obtained by deleting one of the vertices. The following membership property must be satisfied: if σ is in the simplicial complex, then all faces of σ must be in the simplicial complex.

We think of 0-simplices as vertices, 1-simplices as edges, 2-simplices as triangular faces, and 3-simplices as tetrahedra.

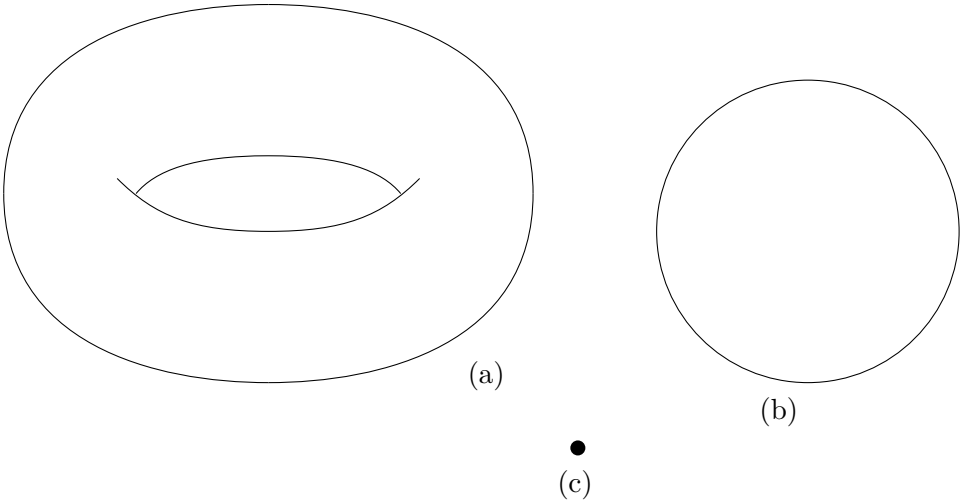


Figure 4.1: ***Elucidation of Betti Numbers***: (a)- 1 connected component, 2 loops and 1 void, (b)- 1 connected component and 1 loop, (c)- 1 connected component

Homology Betti numbers help describe the homology of a simplicial complex X. The value $Betti_k$, where $k \in N$, is equal to the rank of the k-th homology group of X. Roughly speaking, the k^{th} Betti number gives the number of k-dimensional holes. In particular, $Betti_0$ is the number of connected components. For instance, a k-dimensional sphere, has all Betti numbers equal to zero except for $Betti_0$ = $Betti_k$ = 1. In algebraic topology, the Betti numbers are used to distinguish topological spaces based on the connectivity of n-dimensional simplicial complexes. To gain more insight into the concept, see Figure-4.1. Thus, for example, a torus has one connected surface component so $Betti_0$=1, a circular (two-dimensional) hole at each end of the central space, so $Betti_1$=2 and a single cavity enclosed within the surface so $Betti_2$=1. Similarly, for a circle, $Betti_0$ = $Betti_1$ = 1, and for a point (or a filled circle) $Betti_0$ = 1; all higher Betti numbers are zero, for both the circle and the point.

Filtered simplicial complexes A filtration on a simplicial complex X is a collection of subcomplexes $X(t)|t \in R$ of X such that $X(t) \subset X(s)$ whenever $t \leq s$. The filtration time of a simplex $\sigma \in X$ is the smallest t such that $\sigma \in X(t)$. Javapleax is a library that implements persistent homology and related techniques from computational and applied topology, designed for ease of use, ease of access from Matlab and java-based systems, and ease of extensions for further research projects and approaches [67]. In javaPlex, filtered simplicial complexes (or more generally filtered chain complexes) are called streams.

In this work, on the embeddings produced by dimensionality reduction, we perform both hierarchical clustering and compute its persistent homology. The $Betti_0$ of the embeddings produces the same number as the number of relevant clusters

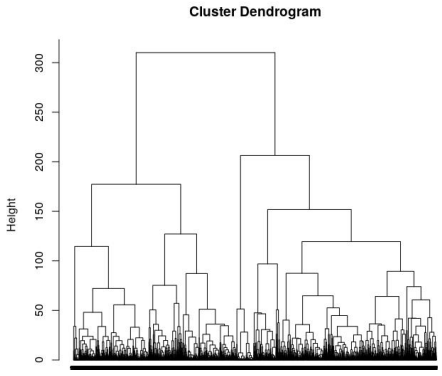


Figure 4.2: *Clustering hierarchy for CDC42*

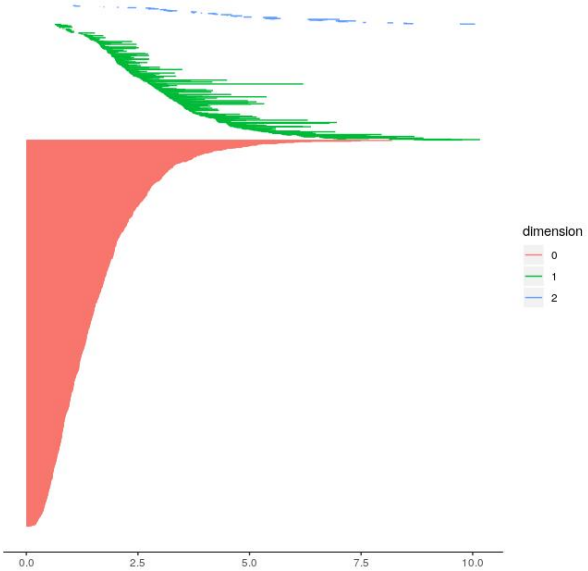


Figure 4.3: *Barcodes of the first cluster for CDC42*

in hierarchical clustering. The results are presented in the next section. Furthermore, these clusters are then subjected to further topological analysis. We used the R package TDA's (Topological Data Analysis) function *calculate_homology* [68]. We say a cluster is relevant if it has at least 100 data instances. Each of these clusters (of a molecule) is a connected component of the embedding. Consequently, the $Betti_0$ of each of these is 1. We observe $Betti_1$ and $Betti_2$ (loops and voids respectively) of each of these clusters to find out whether these clusters are truly distinct conformations of the molecule or not.

In this work we chose a number of medium to large proteins with different conformational dynamics. As mentioned earlier, to generate clusters from the reduced dimension embedding various methods were used depending on the molecular dataset. A thorough analysis for the various molecules is as under.

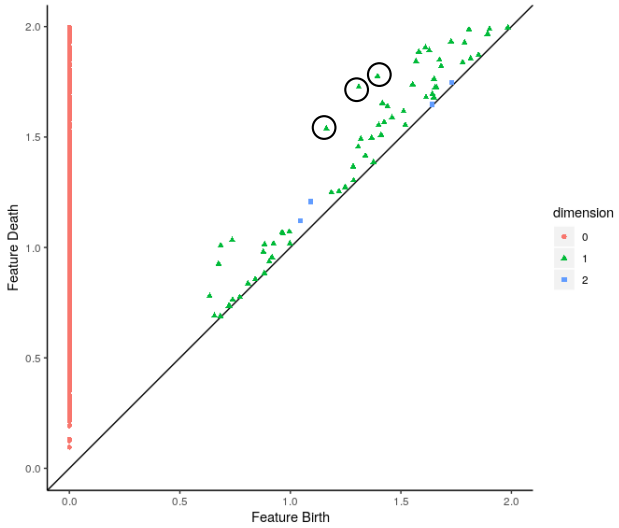


Figure 4.4: *Quantitative representation of the barcodes for the second cluster for CDC42*

4.2.3 CDC42

Cell Division Control-42 is a GTP (Guanosine Triphosphate) binding protein [69]. Human CDC42 is a small GTPase with 191 amino acids of the *Rho* protein family, which regulates signaling pathways that control diverse cellular functions including cell morphology, cell migration, endocytosis and cell cycle progression. It switches between cycles an active GTP-bound state(4did) and an inactive GDP (Guanosine Diphosphate)-bound state (4js0). Recently, CDC42 has been shown to actively assist in cancer progression. Several studies have established the basis for this and hypothesized about the underlying mechanisms [70]. The data for all of the molecules was procured using Molecular Dynamics trajectories.

4DID:
Both forms of hierarchical clustering produced the same clusters here, shown in Figure-4.2. There were four significant clusters produced for 4did. We call a cluster significant if it has at least 100 data instances. Expectedly, when the homology of the entire embedding is computed it returns the $Betti_0$ of 4, meaning there are four connected components in the embedding. These four clusters are then investigated for structural comparison to deduce conformational information. The first cluster's barcodes are depicted in Figure-4.3.

The longer bars are the significant ones, the small bars are considered noise. The figure suggests that there is one connected component with $Betti_0 = 1$, it has a number of loops, denoted by a number of long green bars. The conformation has a number of small voids, denoted by intermittent blue bars. As said earlier,

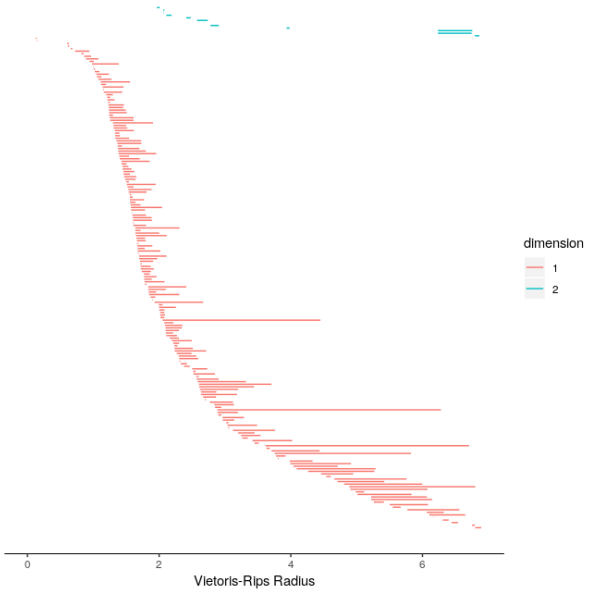


Figure 4.5: *Barcodes of the second cluster for CDC42*

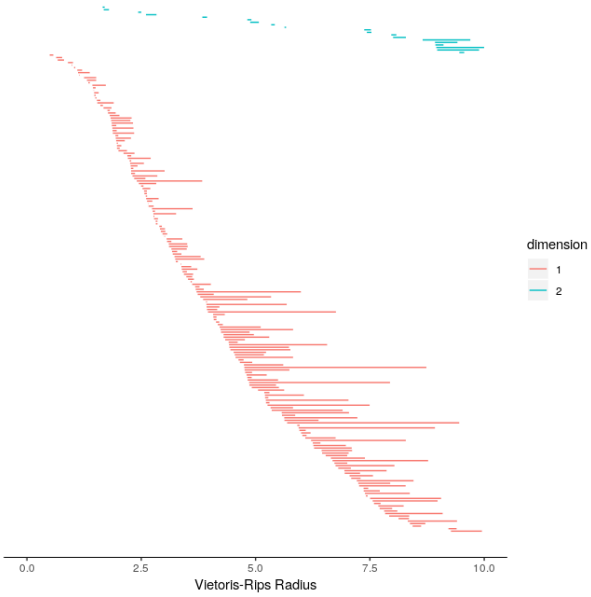


Figure 4.6: *Barcodes of the third cluster for CDC42*

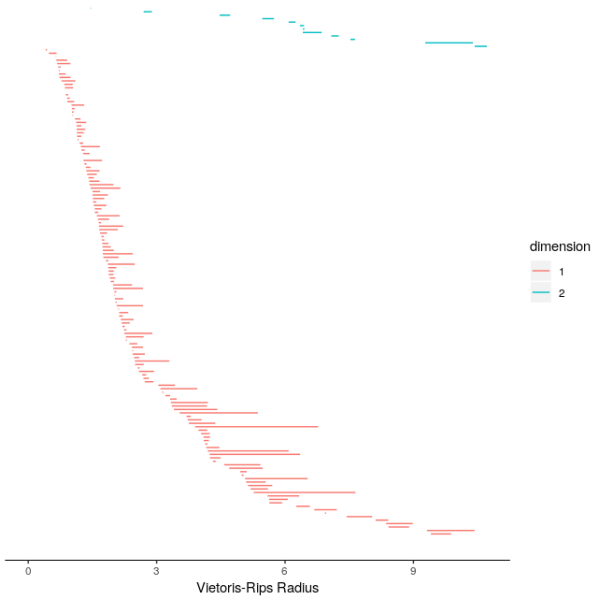


Figure 4.7: *Barcodes of the fourth cluster for CDC42*

all of the clusters (in all the molecules) are one connected component of a larger structure, so $Betti_0$ of all of them is 1. The second cluster, in Figure-4.5, on the other hand has $Betti_1 = 3$, with two significant loops and no significant voids making $Betti_2 = 0$. A quantitative representation of these barcodes for this cluster is shown in Figure-4.4. It is another way of visualizing persistent homology. Each feature is depicted by a single point with the horizontal axis representing feature birth and the vertical axis representing feature death. The line y=x is included as a reference. Since feature birth always precedes feature death (x < y), all points in a persistence diagram lie above the reference line. In persistence diagrams, the persistence of a feature is depicted as the vertical distance between the point representing the feature and the reference line. Just like topological barcodes, feature dimension is coded as the point's color, with an accompanying legend. In the Figure-4.4, we see a number of 0-cycles (red circles); it is difficult to differentiate between most individual 0-cycles due to all of them being a part of the same connected component. However, there a number of loops (blue triangles) two of which have been encircled, they are the ones that correspond to the two loops of this cluster. The upper encircled triangle has a smaller y-coordinate than upper triangle closest to it, yet this is the one that contributes to the $Betti_1$. When evaluating persistence diagrams, it must be kept in mind that a point's y-coordinate by itself is not an accurate measure of feature persistence; rather, it is the vertical distance between the point and the reference line that represents persistence. We count a feature if it is at a distance of at least 1.5 from the reference line. None of the small blue squares make this cut, which makes the $Betti_2$ of this cluster to be

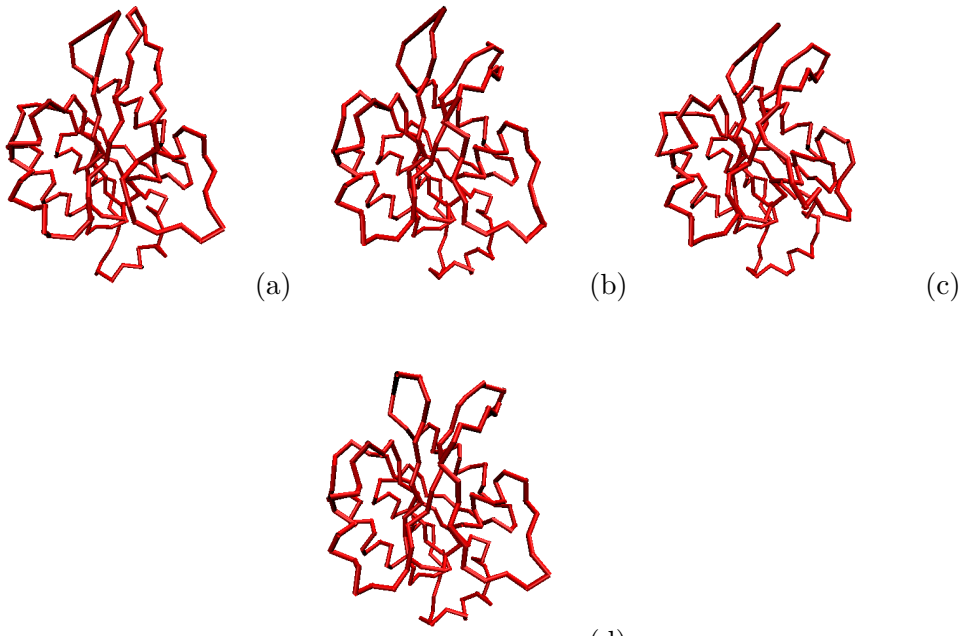


Figure 4.8: ***Conformations of CDC42*:** (a)- Conformation representative of first cluster, (b)- Conformation representative of second cluster, (c)- Conformation representative of third cluster, (d)- Conformation representative of fourth cluster

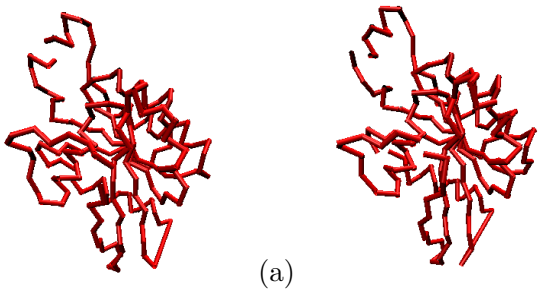
zero. Betti numbers for all other clusters for CDC42 and all other molecules were evaluated similarly. The third cluster, in Figure-4.6, has three small but clearly defined slightly overlapping voids, making $Betti_2 = 3$. Figure-4.7 shows this for the fourth cluster which has a number of overlapping voids only one of which is significant, making $Betti_2 = 1$ here. The significant thing to note here is that all of these conformations have different distributions of the higher Betti numbers which means that conformations represented by these clusters are all different. One conformation representative of each of these clusters is shown in Figure-4.8.

4JS0

This conformation of CDC42 was also subjected to both forms of hierarchical clustering that produced the same set of significant clusters. Here there were just two and both of them are similar. The two conformations representative of this cluster are shown in Figure-4.9.

4.2.4 Oxytocin and Vasopressin

Oxytocin and Vasopressin are both small peptide hormones, both with 9 amino acids[71]. They differ only in two positions in their amino acid sequence, as shown under:



(a) (b)

Figure 4.9: ***Conformations of CDC42-4js0*:** (a)- Conformation representative of first cluster, (b)- Conformation representative of second cluster

Oxytocin: Cys-Tyr-**Ile**-Gln-Asn-Cys-Pro-**Leu**-Gly
Vasopressin: Cys-Tyr-**Phe**-Gln-Asn-Cys-Pro-**Arg**-Gly

These are smaller peptides that do not have one stable 3-D structure. So, intuitively, divisive hierarchical clustering works better to sample conformations here. Nonetheless, for oxytocin we performed agglomerative clustering and obtained five clusters, but their analysis revealed that most of them were way too similar. The first three Betti numbers of these five clusters are in Table-4.1. These numbers indicate that there really are just two different clusters here, which was also the case when a divisive form of hierarchical clustering was performed which resulted in just two significant clusters. The first cluster had only a two small loops and no voids. The second cluster on the other hand had three distinctively large loops and two small voids. Similarly, for Vasopressin, there was just one significant cluster produced by divisive hierarchical clustering. The dendrograms for both oxytocin and vasopressin is shown in Figures-4.10 and 4.11 respectively. The only cluster in Vasopressin is close in structure with the known Oxytocin conformation which was expected given the similarity in the sequences of the amino acids of the two molecules.

Table 4.1: Betti Numbers results for Oxytocin.

Cluster No.	$Betti_0$	$Betti_1$	$Betti_2$
1	1	1	0
2	1	1	0
3	1	1	0
4	1	4	0
5	1	0	0

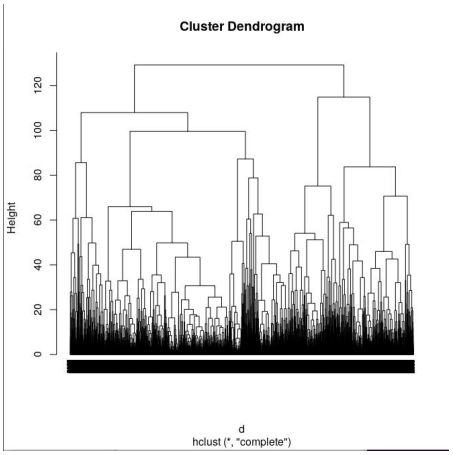


Figure 4.10: *Clustering hierarchy for Oxytocin*

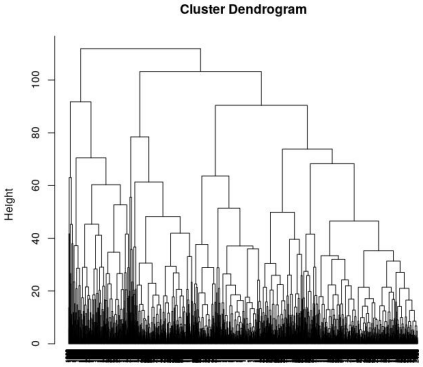


Figure 4.11: *Clustering hierarchy for Vasopressin*

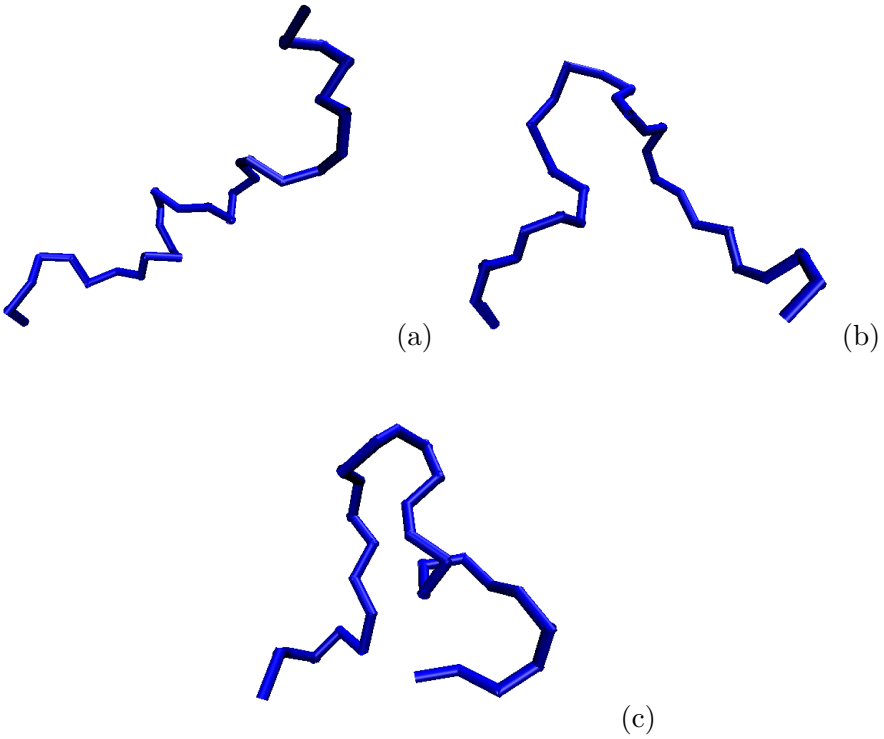


Figure 4.12: ***Conformations of Galanins:*** (a)- Conformation representative of second cluster of hGalanin, (b)- Conformation representative of first cluster of hGalanin, (c)- Conformation representative of the pGalanin cluster

4.2.5 Human and Porcine Galanin

Galanin is a neuropeptide encoded by the GAL gene that is widely expressed in the brain, spinal cord, and gut of humans as well as other mammals [72]. Porcine Galanin (pGalanin) and human Galanin (hGalanin) are two variants of Galanin that differ by 5 out of their amino acids. pGalanin contains 29 amino acids and hGalanin has 30. Both these molecules are also not known to undergo large scale conformational changes. Divisive hierarchical clustering on hGalanin resulted in two major clusters and that of pGalanin produced only one relevant cluster, shown in Figures-4.13 and 4.14. The two structures of hGalanin are quite different from each other, shown in Figures-4.15 and 4.16, which is in agreement with experimental results [73]. The pGalanin barcodes are shown in Figure-4.17. As is evident from the barcodes, the first cluster of hGalanin and the pGalanin cluster are very similar. The study of hGalanin in [73] also shows that there are two molecular forms of hGalanin, one of 30 and another of 19 amino acids. The larger of the two peptides has a sequence which is identical to that of pGalanin except for the following substitutions: Val16, Asn17, Asn26, Thr29 and Ser30. The PDB structures of these clusters is shown in Figure-4.12.

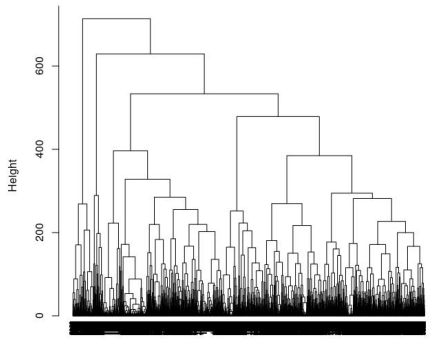


Figure 4.13: *Clustering hierarchy for hGalanin*

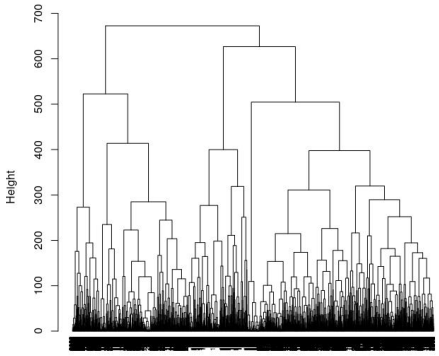


Figure 4.14: *Clustering hierarchy for pGalanin*

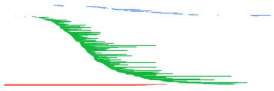


Figure 4.15: *Barcodes for the first cluster of hGalanin*

Figure 4.16: *Barcodes for the second cluster of hGalanin*

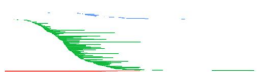


Figure 4.17: *Barcodes for the pGalanin cluster*

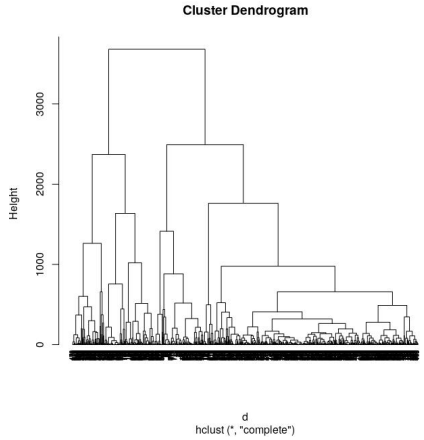


Figure 4.18: *Clustering hierarchy for GroEl*

4.2.6 GroEL

GroEl is a known chaperonin protein which is needed by many other proteins for their proper folding. Structurally, GroEL is a dual-ringed tetradecamer, with both the *cis* and *trans* rings consisting of seven subunits each. The conformational changes that occur within the central cavity of GroEL cause for the inside of GroEL to become hydrophilic, rather than hydrophobic, and is likely what facilitates protein folding. It is required for the correct folding of many proteins. GroEL requires the lid-like co-chaperonin protein complex GroES. Binding of substrate protein, in addition to binding of ATP, induces an extensive conformational change that allows association of the binary complex with GroES. It is the heaviest molecule in the study with 524 amino acids. It is known to undergo large scale conformational changes and hence agglomerative form of hierarchical clustering was used here. It helped in identifying six different clusters, the dendrogram is in Figure-4.18. The conformations representative of each of these clusters are shown in Figure-4.19.

4.2.7 Energy Filtration

As mentioned earlier, the data here was generated using molecular dynamics simulations; they also yield the energy of each simulated conformation. Topological analysis of the entire energy space is the same as computing persistent homology of the entire embedding generated after feature reduction. In this step, we filtered the conformations based on their energy. Figure-4.24 shows the clusters of the 4did version of CDC42 at 100%, 75%, 50% and 25% energy filtration respectively. At k% filtration, the space contains k% lower energy conformations. The colors from blue to yellow signify a drop in energy. As the figures suggest, the increase in the amount of filtration creates voids in the conformational space and most high energy conformations are filtered out (signified by the dominant yellow). At 100%

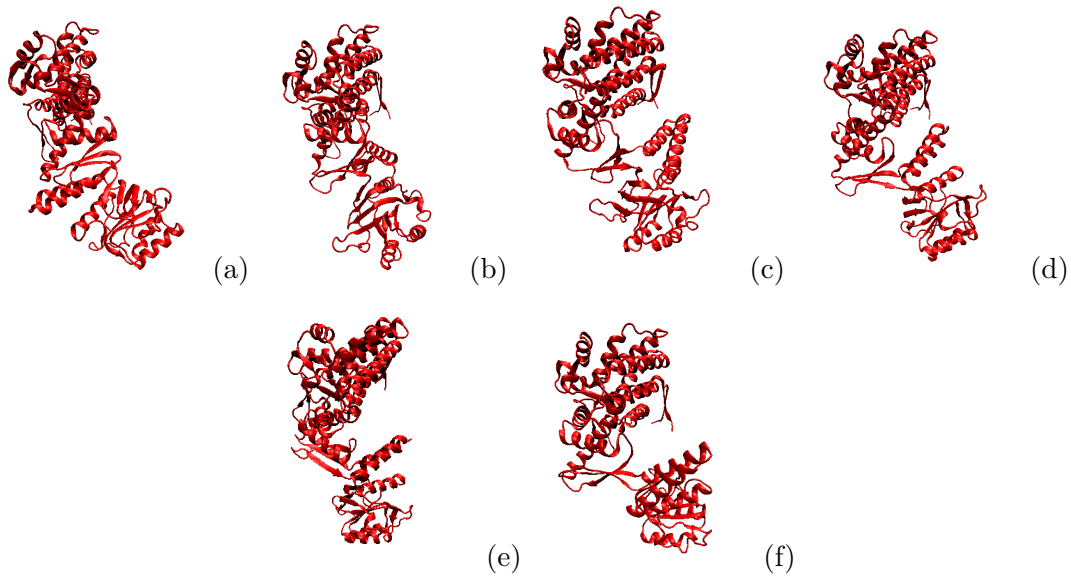


Figure 4.19: ***Conformations of GroEl:*** (a)- Conformation representative of the first cluster, (b)- Conformation representative of the second cluster, (c)- Conformation representative of the third cluster, (d)- Conformation representative of the fourth cluster, (e)- Conformation representative of the fifth cluster, (f)- Conformation representative of the sixth cluster

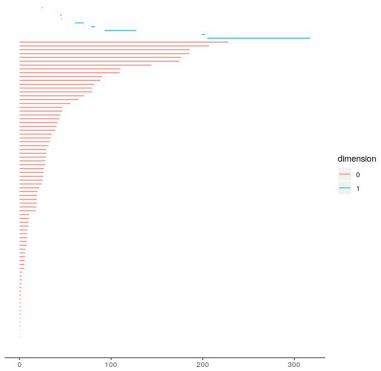


Figure 4.20: *Barcodes for the first cluster of GroEl.*

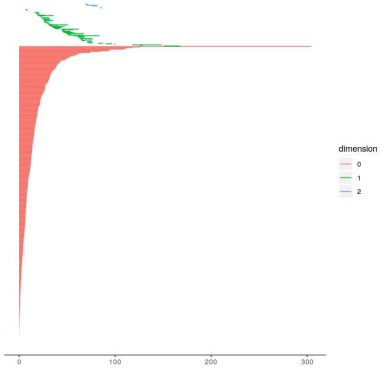


Figure 4.21: *Barcodes for the second GroEl cluster.*

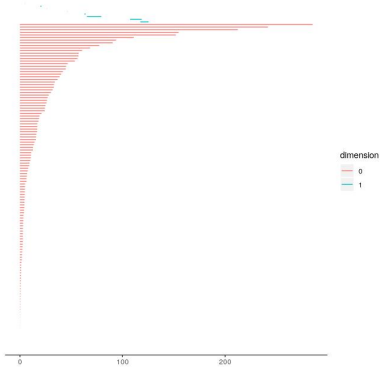


Figure 4.22: *Barcodes for the third cluster of GroEl.*

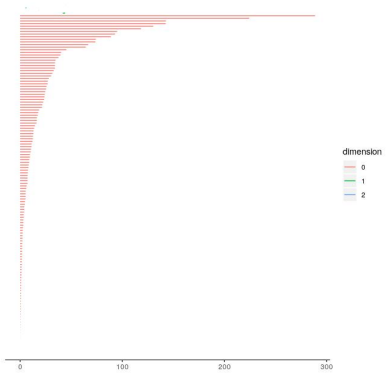
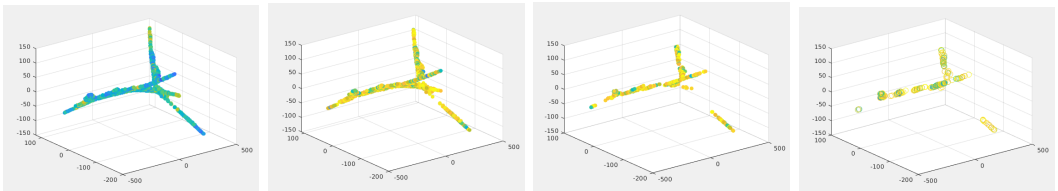


Figure 4.23: *Barcodes for the fourth cluster of GroEl.*



(a) (b) (c) (d)

Figure 4.24: ***Energy clusters of 4did:*** *(a)The entire energy space, (b) 75% energy space, (c) 50% energy space, (d) 25% energy space*

filtration, the entire space is represented as one connected component (Figure-4.24(a)). At 75%, the high energy conformations, represented by blue, have been removed and a small void appears between components that are near. There are significant changes in the embedding at 50% filtration. The persistence diagrams are shown in Figure-4.25. At 25% filtration, number of connected components are maximum and also the number of loops, shown in Figure-4.25(c). The number of loops here is the sum of the loops in all the connected components. The four clusters produced by algebraic topology all have energy below 50%, and three of them have energy below 25%. This is proof that effective dimensionality reduction followed by algebraic topology coupled with hierarchical clustering produces clusters of conformations that are probable and comparatively low energy.

Similarly, energy filtration of 4js0 at various levels of filtration is shown in Figure-4.26. At 75% filtration(not shown in figure), the space looked very similar to that of part (a) of the figure. Even at 50% filtration, the structure of the embedding is retained with two large connected components. At 25% filtration, there survived only one significant connected component with a number of loops in the entire structure. The two conformations of this molecule, isolated by algebraic topology above, both have energy less than 50%, which is the average for all the conformations generated by the simulation. One of these two, belong to the set of conformations with less than 25% energy.

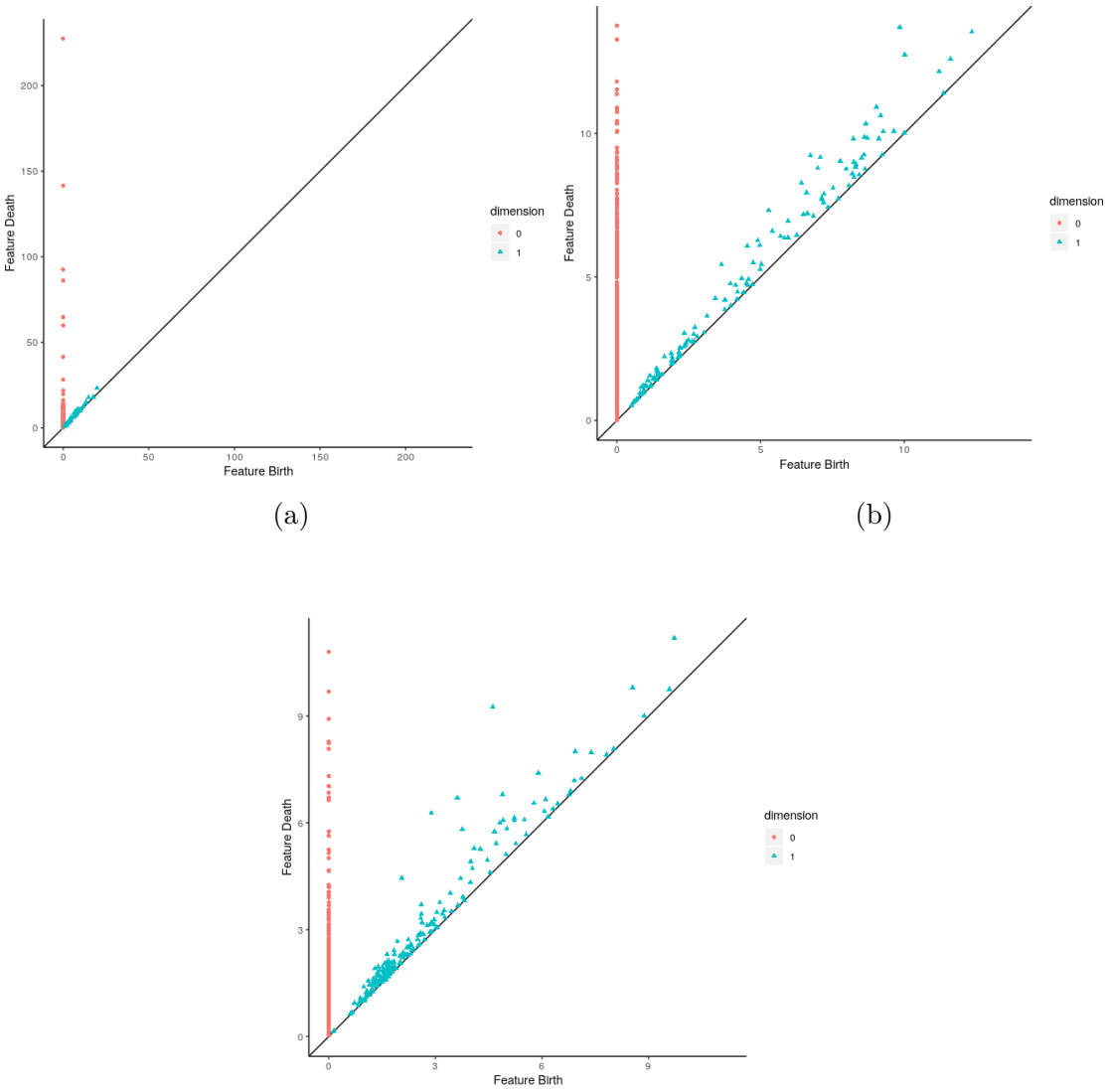
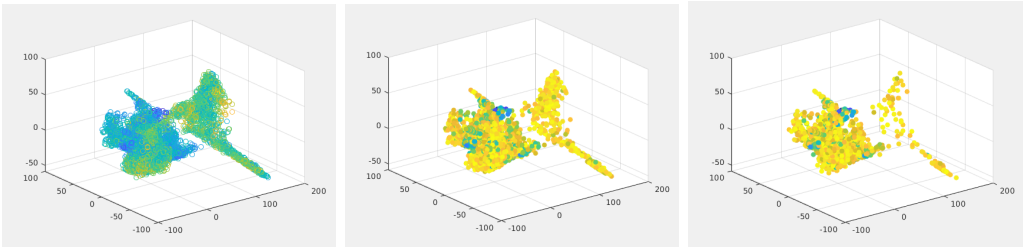


Figure 4.25: ***Persistence diagram for 4did:*** *(a) at 75% energy filtration, there are four relevant connected components, (b) at 50% filtration, three relevant clusters with a number of loops, (c) at 25% filtration with increased connected components, and an increase in number of loops.*



(a) (b) (c)

Figure 4.26: ***Energy clusters of 4js0:*** *(a) The entire energy space, (b) at 50% filtration, (c) at 25% filtration.*

Chapter 5

IMPROVED SEARCH METHODOLOGIES FOR SIMULATING PROTEIN CONFORMATIONAL PATHWAYS

The work described in this chapter is based on a preliminary work presented in [61]. The problem addressed in this work is to extend our structural characterization to dynamic proteins that switch between different structural states to modulate their biological function. Specifically, we propose a way to compute molecular motions employed by dynamic proteins in switching between different functionally relevant structural states.

5.1 Problem Statement

Given a protein that undergoes a large scale conformational motion, we present an algorithm to search for conformational pathway that link the two end-points. The input to the algorithm is the two end-points, denoted start and end, and the output is a trajectory made of conformations along the pathway. The degrees of freedom of the protein molecule are defined as the dihedral angles or planar angles located between various residues. While usually secondary structure elements do not change much during conformational transition and most of the protein flexibility lies in loops, this is not always the case. Each conformation in the sampling path is ranked with a potential energy and is retained if its value is lower than a threshold. The path needs to satisfy additional constraints. The energetic difference between conformations in a path is controlled via different means, either by placing a bound on the maximum energy in a path or through the Metropolis criterion.

5.2 Related Work

The conformational changes are usually a continuous and transient process and intermediate structures are hard to be determined experimentally. Nonetheless, the intermediate structures are important for our understanding of the mechanism by which proteins interact with ligands or other proteins. They help us to design drugs targeting proteins and understand critically important processes in the cell. Computational methods provide an efficient way to calculate conformational changes in proteins and other macromolecules. Molecular Dynamics (MD) simulations [5] can now sample protein dynamics up to the microsecond level but they are still not fast enough to capture conformational motions that take place over larger timescales – such as folding or conformational changes that may take place over milliseconds or more. Another class of methods sample the conformational space efficiently by perturbing the molecular structure at certain degrees of freedom and using geometric and biophysical constraints to guide the search. The search is done using various techniques such as normal mode analysis (NMA) [13, 11], elastic network [74, 75], robotic motion [76, 77] planning and more. These methods are usually faster than MD but do not result in physical pathways due to randomness in the search and the approximations being used. Therefore, further processing should often be applied to extract biologically meaningful information. Several motion-planning methods were developed in the past to sample the conformational space of proteins using prior knowledge [8] or multi-scale sampling [10, 7, 78]. In this work, we present a biased Monte Carlo (MC)- based algorithm called Rapidly-exploring random tree (RRT) to efficiently sample the conformational pathways on medium and large sized proteins.

5.2.1 Molecular Dynamics based Approaches

Many adaptations are pursued to lower the computational demands of MD-based approaches. Essentially, MD-based methods for elucidating transitions incorporate some suitable bias at the expense of obtaining possibly different transition trajectories. Methods include targeted, biased, or steered MD, importance sampling, umbrella sampling, replica exchange, local flattening of the energy surface, activation relaxation, conformational flooding, swarm methods, and others [79]. Efficiency concerns are also addressed through coarse graining and techniques based on normal mode analysis and elastic network modeling. Some methods focus on deforming a trivial conformational path (obtained, for instance, through morphing) to improve its energy profile. Examples include the nudged elastic band, morphing, zero-temperature string, and finite-temperature string methods. While the incorporation of a suitable bias towards the goal structure forces the simulation to reduce dwell time in a given stable or meta-stable state, the bias possibly sacri-

fices a more expansive view of possibly different transition trajectories to the goal structure. This is typically addressed by repeating the simulation to obtain many transition trajectories, which taken together can cover the transition ensemble in the absence of correlations between trajectories.

5.2.2 Robotics Inspired Tree-based Approaches

Since simulation of dynamics is the limiting factor in dynamics-based methods, efficiency concerns can be addressed by foregoing or at least delaying dynamics until credible conformational paths have been obtained. A different class of methods focus not on producing transition trajectories but rather computing a sequence of conformations (a conformational path) with a credible energy profile. The working assumption is that credible conformational paths can then be locally deformed with techniques that consider dynamics to obtain transition trajectories. Most notably, methods in this category adapt sampling-based search algorithms developed for the robot motion-planning problem which bears strong analogies to the problem of computing conformations along a structural transition. The objective in robot motion planning is to obtain paths that take a robot from a start to a goal configuration. In robotics and molecular motion computation, a start and a goal state are specified. The goal is then to produce a feasible path that the system can follow to navigate its environment and transition from the start to the goal state. There are some unique challenges offered up by molecular systems. First, molecular systems typically have an exceptionally high number of degrees of freedom. Also, in a protein system, the cost surface or energy surface is typically more complex and with many local minima. The methods developed in algorithmic robotics to address the robot motion planning problem fall under either the roadmap-based or tree-based category. The method we propose in this chapter falls under tree-based methods. Roadmap-based methods suffer from what is called the *steering problem*, which means, if one has two sampled conformations or configurations, it may not be possible to find a constraint-satisfying path steering the system from one conformation to another. Tree-based methods have a number of ways to guide the search, they are broadly classified into, expansive search trees(EST) [80], path- directed subdivision tree (PDST) [81] and, the method followed in this work, rapidly expanding random tree (RRT). It is discussed in detail in the next section. All these approaches vary by how the tree is grown in the robot configuration space. EST expands the tree by selecting and expanding existing configurations within the tree. Briefly, a configuration is selected using some probabilistic weighting scheme. The selection configuration is then slightly perturbed to arrive at a new configuration. If a collision free path can be obtained between the selection and the new configuration, the new configuration can be added to the tree. PDST, on the other hand, was designed to deal with motion problems that suffer from

significant drift, under actuation, and discrete system changes. This method uses a low-dimensional projection of the sampled configurations to approximate coverage of the configuration space. This projection is decomposed into cells. All cells are stored in a priority queue based on a score. The expansion of the tree proceeds as follows. The highest scoring cell is dequeued, and a configuration from the cell is selected uniformly at random. This selected cell is then subdivided and the resulting cells are returned to the priority queue. A cell's score is primarily based on its size, with larger cells given higher priority. A perturbation function is then applied to the selected configuration resulting in a new configuration which is stored in the tree and mapped to its appropriate cell. By selecting larger cells, the search is biased towards under explored areas of configuration space.

5.3 Rapidly-expanding Random Tree

A rapidly exploring random tree (RRT) is an algorithm designed to efficiently search non-convex, high-dimensional spaces by randomly building a space-filling tree. The tree is constructed incrementally from samples drawn randomly from the search space and is inherently biased to grow towards large unsearched areas of the problem. RRTs were developed by Steven M. LaValle and James J. Kuffner Jr [82]. RRTs can be viewed as a technique to generate open-loop trajectories for nonlinear systems with state constraints. An RRT can also be considered as a Monte-Carlo method to bias search into the largest *Voronoi regions* [1] of a graph in a configuration space.

The conformational changes are usually a continuous and transient process and intermediate structures are hard to be determine experimentally. Nonetheless, the intermediate structures are important for our understanding of the mechanism by which proteins interact with ligands or other proteins. They help us to design drugs targeting proteins and understand critically important processes in the cell. Following is a brief account of various methods to do so and has been adapted from [83]. Computational methods provide an efficient way to calculate conformational changes in proteins and other macromolecules. Molecular Dynamics (MD) simulations [5] can now sample protein dynamics up to the microsecond level but they are still not fast enough to capture conformational motions that take place over larger timescales such as folding or conformational changes that may take place over milliseconds or more. Another class of methods sample the conformational space efficiently by perturbing the molecular structure at certain degrees of freedom and using geometric and biophysical constraints to guide the search. The search is done using various techniques such as normal mode analysis (NMA)

[1] In mathematics, a Voronoi diagram is a partitioning of a plane into regions based on distance to points in a specific subset of the plane.

[13, 11], elastic network [74, 75], robotic motion [76, 77] planning and more. These methods are usually faster than MD but do not result in physical pathways due to randomness in the search and the approximations being used. Therefore, further processing should often be applied to extract biologically meaningful information. Several motion-planning methods were developed in the past to sample the conformational space of proteins using prior knowledge [8] or multi-scale sampling [10, 7, 78]. Another sampling based motion planning approach is described in [84]. A brief review of adaptive sampling methods is in [85]. In [86], the authors presented a hierarchical clustering and algebraic topology based method on coarse grained protein conformational search that detects regions of interest in protein conformational space on. An accurate sampling of protein trajectory helps better identifying intermediate clusters which also correspond to locally minimal conformations. The SPIRAL method described in [87] tries to solve the same problem of simulating molecular motion but is computationally expensive. A novel evolutionary algorithm that aims to sample regions of interest in the conformational space is presented in [88]. It presents a promising research direction in improving decoy generation for template-free protein structure prediction. The same problem is addressed in [89] by using a highly parallelized version of RRT called, Transition-based RRT. This algorithm also aims at identifying most probable conformations of a class highly-flexible biomolecules such as Intrinsically Disordered Proteins (IDPs).

Many methods attempt to lower the computational demands of MD-based approaches. Essentially, MD-based methods for elucidating conformational transitions incorporate some suitable bias at the expense of obtaining possibly different transition trajectories. Methods include targeted, biased, or steered MD, umbrella sampling, replica exchange, activation relaxation, conformational flooding, swarm methods, and others [79, 90, 91, 92]. Efficiency concerns are also addressed through coarse graining and techniques based on normal mode analysis and elastic network modeling. Some methods focus on deforming a trivial conformational path (obtained, for instance, through morphing) to improve its energy profile. In [10], authors combined methods originating from robotics and computational biophysics, to model protein conformational transitions. To reduce the dimension of the problem, normal modes are computed for a coarse-grained elastic network model built on short fragments of three residues. This added biased helps to efficiently sample conformational search space. While the incorporation of a suitable bias towards the goal has an impact on the accuracy of the simulation, structure forces reduce dwell time in a given stable or meta-stable state, the bias possibly sacrifices a more expansive view of possibly different transition trajectories to the goal structure. This is typically addressed by repeating the simulation to obtain many transition trajectories, which taken together can cover the transition en-

semble in the absence of correlations between trajectories. A brief review of the current methods and their comparison can be found in [93].

A different class of methods focus not on producing transition trajectories but rather computing a sequence of conformations (a conformational path) with a credible energy profile. The working assumption is that credible conformational paths can then be locally deformed with techniques that consider dynamics to obtain transition trajectories. Most notably, methods in this category adapt sampling-based search algorithms [94] developed for the robot motion-planning problem which bears strong analogies to the problem of computing conformations along a structural transition. There are some unique challenges offered up by molecular systems. First, molecular systems typically have an exceptionally high number of degrees of freedom. Also, in a protein system, the cost surface or energy surface is typically more complex and with many local minima. The methods developed in algorithmic robotics to address the robot motion planning problem fall under either the roadmap-based or tree-based category. Roadmap-based methods [87] are limited by the *steering problem*, which means, if one has two sampled conformations, it may not be possible to find a constraint-satisfying path steering the system from one conformation to another. This method uses a low-dimensional projection of the sampled conformations to approximate coverage of the conformational space.

5.3.1 Evolutionary Information from Multiple Sequence Alignment

Over the past few years, there has been a growing interest in methods that apply evolutionary information extracted form protein sequence changes to protein structural and functional problems such as protein folding and protein-protein interactions, and identifying proteins hotspot residues [95, 96, 97, 98]. Hotspot residues help in stabilizing the protein during conformational changes [99]. Since sequence information is much more readily available than structural information, many methods attempt to identify the co-evolving residues within a protein using multiple sequence alignment (MSA). Existing approaches use various methods of statistical analysis to detect signals among co-evolving residues. These methods utilize phylogenetics, undirected graphical models, deep learning, and more [98, 96]. This information has been used to reduce the search space on multiple domains of structural bioinformatics. In this work, we present a biased Monte Carlo tree search (MCTS)-based algorithm to efficiently sample the conformational trajectory on medium and large sized proteins. Given a protein that undergoes a large scale conformational motion, we present an algorithm to search for conformational trajectory that links the two end-points. The input to the algorithm is the two end-points, denoted as start and end conformations, and the output is a trajectory made of conformations along the pathway.

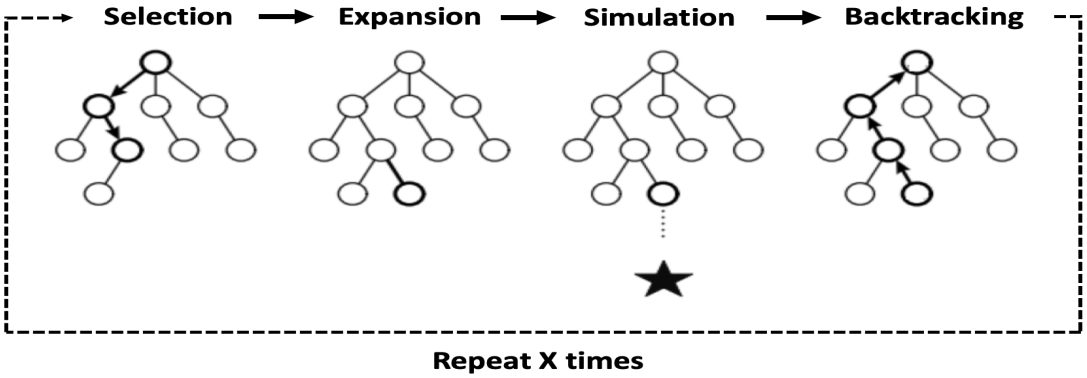


Figure 5.1: *An overview of steps of the Monte Carlo tree search algorithm*

5.3.2 Simplifying the Search Space

We begin with the start and goal (end points) conformations of a protein. These two endpoints are the structures of the molecule, where the molecule undergoes transformation from the start to the goal. The degrees of freedom of the protein molecule are defined as the dihedral angles or spatial angles located along the backbone, as we consider bond lengths and angles fixed. We decide upon a score threshold (optimal RMSD) which is the acceptable distance tolerance between the actual goal conformation and the one discovered by the search. We run the algorithm until the threshold is reached, or until a maximum level of iterations is run.

The resolution of proteins representation of our algorithm is on backbone + $C-\beta$. We represent every conformation with two lists of the backbone dihedral angles ϕ and ψ. ϕ involves the backbone atoms $C-N-C_\alpha-C$, and ψ involves the backbone atoms $N-C_\alpha-C-N$ between two consecutive residues. All the perturbation are performed on the backbone dihedral angles.

5.3.3 Search Methodology

The conformational pool between two endpoints is generated by a biased Monte Carlo tree search (MCTS) algorithm. Figure 5.1 represents an overview of MCTS steps. The conformation pool contains initially only the start conformation. Each iteration is divided into four steps and the pool keeps expanding. During the selection step, a conformation called the successive conformation (node) is selected for the expansion. Initially, the algorithm starts with one of the given endpoints which is denoted by start conformation. At the expansion step for the selected conformation, a residue with the largest difference in dihedral angle (first we consider ϕ and then ψ) compared to the goal conformation is determined. Then the selected residue is randomly perturbed (rotated) by ± 5 degrees and the new conformation is generated. Then the new conformation is corrected for steric clashes

at the atoms level by correcting the distance between two adjacent atoms caused by numeric error accumulated during sampling. The new conformation is added to the pool if it passes the energy criteria. Otherwise the expansion step is repeated by selecting the next largest difference between selected conformation and goal conformaion. The expansion step will be repeated until a new valid conformation is generated. The potential function we used for the semi-coarse grained (backbone $+ C - \beta$ model) is as follows [100]:

$$E_{total} = E_{VdW} + E_{HB} + E_{burial} + E_{water} + E_{bond} + E_{angle}$$

where the VdW, bond and angle terms are taken from the $AMBER\ ff03$ force field [101]. The VdW term was slightly modified to allow soft collisions between atoms, due to the lower resolution. This function was also used by us in the past [102] to cluster conformational changes in proteins. The potential energy threshold for the new perturbed molecule is equal to the number of residues of that molecule (The total potential energy of a new molecule must below than number of its residues).

Once the expansion step is passed and the new selected conformation is added to the conformational pool, the algorithm moves to the simulation step. The goal of the simulation step is to generate a path of conformations between the expanded and the goal conformations based on the following criteria:

1. $lRMSD_{new} < lRMSD_{parent}$
2. $r < e^{-\frac{lRMSD_{new} - lRMSD_{parent}}{lRMSD_{new} \cdot a}}$

Here, r is a random number within the range of [0,1), and a is a predefined constant. This allows us to accept, with small probability, conformations that do not lead us closer to the goal structure, in order to escape possible local minima. In this study, a is defined as 0.01. The sampling continues until $lRMSD_{new}$ is less than a predefined threshold value or the $lRMSD_{minima}$ of the entire pool does not decrease for M iterations. In this study the threshold is set to 1.5 and M is set to 500. Finally, the algorithm gets into backpropagation step if it fails to find a new acceptable path after 1000 attempts. In this case the expanded conformation will be marked for not being selected for the next iteration of MCTS. In order to avoid expanding the conformational pool to some intractable search space, we introduced a guided selection step which occurs every three iterations. In other words, after two consecutive unsuccessful iterations of MCTS, during the selection step of the 3^{rd} iteration of MCTS, we select the successive conformation among the top N conformations with the smallest $lRMSD$ compared to the goal conformation from the entire pool where N is user-defined, and in this work is set to 20.

Eventually a pathway, or multiple pathways, from the tree root to goal con-

formation are detected and the search is finished. Each search can result in a possible pathway, but due to the inherent randomness it may not be the only possible pathway. Combining the results from multiple runs provides an estimation of the conformational space between start and goal protein structures.

5.3.4 Incorporating Evolutionary Information to The Search Space

In a previous work [61], the search space was reduced by perturbing only the dihedral angles that do not correspond to a *rigid* [2] part of the molecule. In this work, we use co-evolution information that can be obtained form MSA. We limit the search space of selecting residues for perturbation to those residues with high co-evolution score. To isolate these specific sites in a molecule, we utilize the PSICOV and Meta-PSICOV methods described in [96, 103]. PSICOV is a probabilistic graphical model-based method for identifying proteins' 3D structure by inferring top co-evolving residues from multiple sequence alignment (MSA) of homologue sequences of the query protein. Meta-Psicov utilizes two stages of the neural network to improve upon PSICOV. In the first stage, the results of PSICOV are combined with GREMLIN [97] to predict the top co-evolving residues within a protein. Later, in the second stage, the predictions are refined based on other features such as amino acid propensity, hydrogen bonds, and secondary structure [103]. Meta-PSICOV returns a list of top co-evolving residues. For a molecule with k residues,we take the $\frac{k}{4}$ top pairs, which at most gives $\frac{k}{2}$ residues of interest on the molecule, thereby reducing the search space by half (some residues may appear among top co-evolving pairs of residues more than once with different partners). After identifying these residues from Meta-PSICOV, this information integrated into the MCTS method during the expansion step where the selection of the residues for the perturbation is limited to those residues that have a high co-evolution score. Although not all the co-evolving residues that obtained from a MSA corresponding to the hotsplot residues due to phylogenetics and MSA construction biases, this information can help for selecting nearby hotspot residues which may contribute to the stability of proteins conformation.

5.4 Results and Discussion

Table-5.1 shows similar results but without any information about contact-residues. As expected, the time consumed to attain similar results is much more. The molecules used in this study are well known and have concerted structures. They are explained in brief in the section to follow.

[2] The rigid parts of a molecule are its portions that do not undergo any structural change in its course of conformational change.

Table 5.1: RMSD Results with entire search space

Molecule	Conformations	RMSD at start	Accuracy achieved	Time (in min)
Calmodulin	$1CLL \longrightarrow 1CTR$	14.83Å	3.5Å	≈ 55
	$2F3Y \longrightarrow 1CFD$	10.61Å	3.35Å	≈ 63
Adenylate Kinase	$1AKE \longrightarrow 4AKE$	7.13Å	2.8Å	≈ 70
GroEl	$1SS8 \longrightarrow 1SX4$	12.21Å	5.5Å	≈ 65
Chemotaxis	$3CHY \longrightarrow 1CHN$	5.61Å	4.2Å	≈ 120
Cynovirin-N	$2EZM \longrightarrow 1L5B$	13.26Å	4.98Å	≈ 90
Ribose binding protein	$1URP \longrightarrow 2DRI$	4.08Å	2.2Å	≈ 190

5.4.1 Chemotaxis Protein CheY

CheY is a single domain protein, which is a variant of the ubiquitous receiver domain that is found in response regulators of classic two-component signal transduction systems as well as chemotaxis systems. The CheY proteins are cytoplasmic proteins that are part of a signal transduction pathway between the attractant or repellent and the flagellar motor switch. CheA is a histidine kinase which belongs to the class of signaling proteins. It has conserved amino acid sequences at the C-terminal domain which is phosphorylated at a histidine residue, and transfers the phosphoryl group to a response regulator protein. CheY and CheB are response regulator proteins. They have N-terminal domains similar to the other response regulators and undergo phosphorylation and dephosphorylation. CheY transitions between 1CHN and 3CHY with an RMSD of 5.61Å between the two endpoints. The proposed algorithm reaches the goal conformation (3CHY) within accuracy of 4.2Å$\pm$10%. Alluding to the randomness in producing the next conformation to expand, in 10 runs of the algorithm the average RMSD of 4.2Å is plateaued. The runtime for each run is reduced to about half, with the contacting residue directive for the same accuracy. Figure-5.2 shows the contact residues over the backbone of this protein.

5.4.2 Ribose Binding Protein (RBP)

Conformational changes are necessary for the function of bacterial periplasmic receptors in chemotaxis and transport. Here, we used two conformations of the protein from *Eschericia Coli (E.coli)*. In E. coli RBP (Ribose Binding Protein) small-scale backbone movements are restricted to the hinge region, whereas the secondary structure elements in the two domains and the amino acids in the binding pocket adopt essentially the same conformations. According to the studies, [104, 105], the least open form of an RBP is probably the dominant one in solution under normal conditions, although a mixture of species seems likely. The open (1URP) and closed (2DRI) forms have distinct surfaces in the regions known to be important in chemotaxis and transport, which will differentiate their interactions

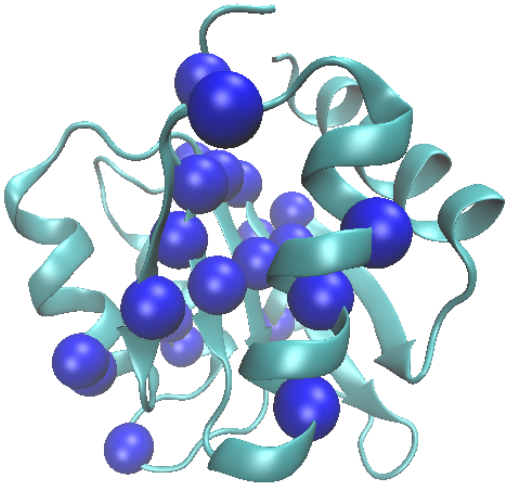


Figure 5.2: *Contact residues of CheY*

with the membrane components. It seems certain that the conformational path that links the forms described here is that followed during ligand retrieval, and in ligand release into the membrane-bound permease system. In this work we try to simulate the conformational change between these two forms and the results are summarized in Tables-5.1 and 5.2. Figure-5.3 shows the contact residues and figures-5.4(a) and (b) show the two extreme conformations of this molecule.

5.4.3 Adenylate Kinase (AdK)

AdK is a monomeric phosphotransferase enzyme that catalyzes reversible transfer of a phosphoryl group from ATP to AMP. The structure of AdK, which contains 214 amino acids, is composed of the three main domains, the CORE (residues 1-29, 68-117, and 161-214), the ATP binding domain called the LID (residues 118-167), and the NMP binding domain (residues 30-67). AdK assumes an 'open' conformation in the unligated structure and a 'closed' conformation. The lRMSD between the two structures is 7.13Å. Supposedly, during the transition from the 'open' to 'closed' form, the largest conformational change occurs in the LID and NMP domain with the CORE domain being relatively rigid. The distance measure threshold for successful termination of the algorithm was a conformation with an RMSD difference of 2.2Å from the goal conformation.

5.4.4 GroEl

The GroEL protein belongs to the chaperonin family and is found in a large number of bacteria. It is required for the correct folding of many proteins. GroEL requires the lid-like co-chaperonin protein complex GroES. Binding of substrate protein, in addition to binding of ATP, induces an extensive conformational change that

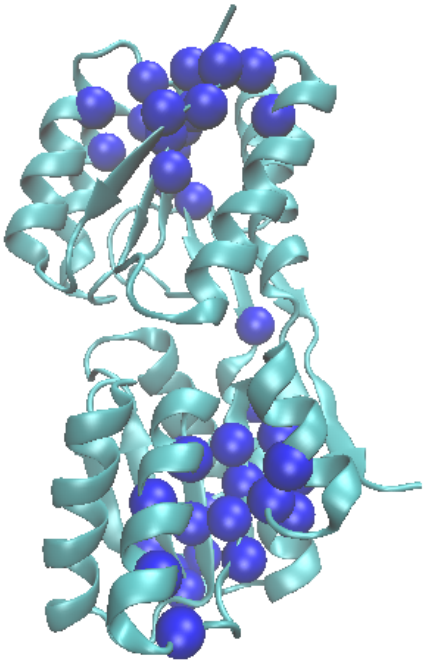


Figure 5.3: *Contact residues of the Ribose-binding protein*

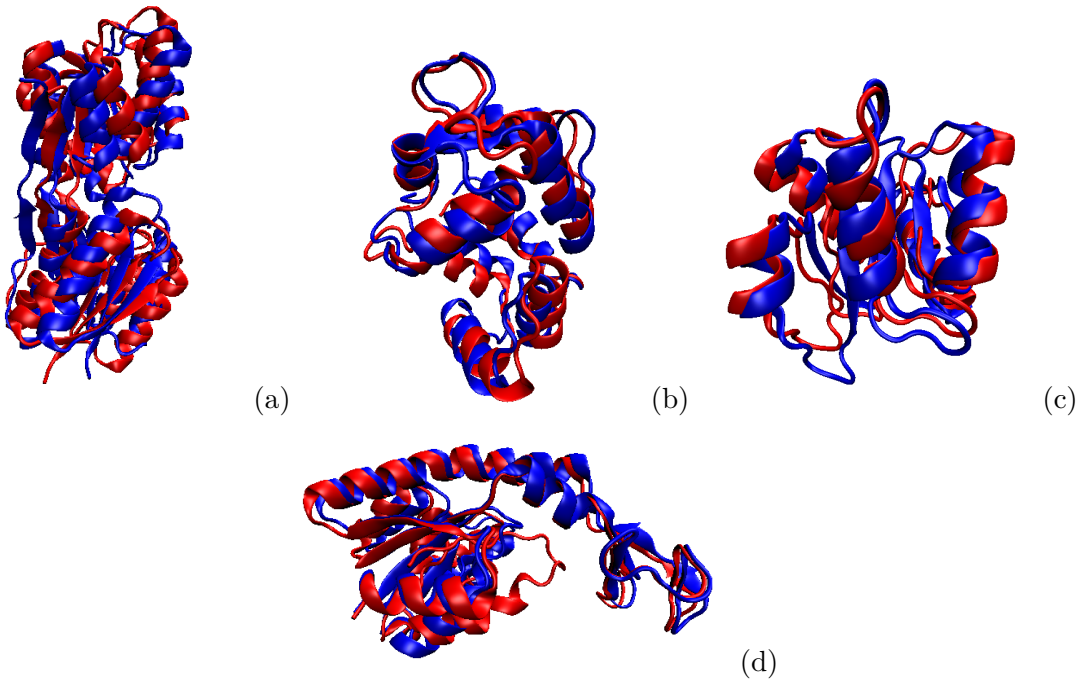


Figure 5.4: *Superimposition of the actual and simulated goal structure: (a) Ribose binding protein-2DRI, (b)Calmodulin- 1CTR, (c)Chemotaxis-1CHN*

allows association of the binary complex with GroES. This was the largest molecule in the study with 524 amino acids. We work with two well-studied conformations of the molecule 1SS8 and 1SX4. The RMSD difference for successful termination of the algorithm was 2.96Å from the goal conformation.

5.4.5 Calmodulin

Calmodulin (CaM) is a multi-functional intermediate calcium-binding messenger protein expressed in all eukaryotic cells. It is an intracellular target of the secondary messenger Ca^{2+}, and the binding of Ca^{2+} is required for the activation of calmodulin. Calmodulin is a small, highly conserved protein that is 148 amino acids long. The protein has two approximately symmetrical globular domains each containing a pair of EF-hand[3] motifs (the N- and C-domain) separated by a flexible linker region for a total of four Ca^{2+} binding sites. The two forms of Calmodulin used for this study are 1CLL and 1CTR, the RMSD difference between them is 14.83Å. The binding site directive produced a goal conformation that was 3.2Å to the actual goal conformation, the regular RRT-MC algorithm without any information on contact residues did so with 3.5Å. The time consumed although was about half with the selective residue directive.

Two other conformations of Calmodulin, namely 1CFD and 2F3Y were also studied. Former is the calcium free form of Calmodulin and the latter is bound to a putative IQ motif in the C-terminal tail of the pore-forming subunit. Figures-5.4(c) and (d) show the two end points of conformations for this molecule.

5.4.6 Cynovirin-N

CVN is an anti-viral fusion inhibitor protein that binds to viral sugars, and is trialed for preventing sexual transmission of HIV. It comprises two repeat domains of 30% sequence identity. The domain swapped dimer has higher anti-viral affinity than the monomer, and it was shown that the two forms can exist in solution, with a high energy transition barrier between them. In addition, it has been reported that certain mutations can affect the energy barrier and stabilize alternative conformations. We simulated the unpacking of the repeat domains of a single chain from the intertwined monomeric conformation to an extended domain-swapped conformation. The two conformations differ form each other with an RMSD of 13.26Å. CVN contains 101 amino acids. As expected, limiting the search space for the algorithm by incorporating the binding site information, the algorithm consumed much less time, with slightly better accuracy towards goal conformation (Table-5.1, 5.2). Figures-5.4(g) and (h) show the two conformations path between

[3]The EF hand is a helix-loop-helix structural domain found in a large family of calcium-binding proteins.

which is approximated.

5.4.7 Analysis of Results

As expected, the results qualitatively represented a path between the two endpoints of their respective proteins. It should be noted that such information is hard to determine with experimental methods due to the transient nature of intermediates. Also, despite the fact that the proteins were represented in a coarse-grained manner and the intermediates were not subjected to further energy minimization, the structures were reasonably close to the actual known structures of the molecule. Table 5.2 shows the results of this study. It reports the RMSD between simulated goal conformation attained by the search and the actual goal conformation, which is the right hand side of the column named Conformations. The algorithm starts with the two end points and the score threshold RMSD that would be acceptable between the actual and simulated goal. So, when an extremely low threshold is chosen, for example 1Å, the simulation runs for longer trying to better the simulated goal towards the actual one. As seen, we were able to generate pathways within low lRMSD with respect to the goal structure (the best lRMSD per pathway was less than 3Å) in nearly all cases. The average time a simulation takes is between 45 minutes to 1 hour, depending on the size of the protein. A conformation close to the goal was found in each run. The method produced thousands of conformations before the final path was detected. Due to the inherent randomness, the pathways produced are not necessarily the biological pathway. The method described in [87] achieves better accuracy and uses the entire molecule for simulation but is computationally more expensive, taking a minimum of 56 CPU hours.

Figure 5.5 shows the RMSD trend of the simulation with (red) and without (blue) the contact-residue directive. The results are significantly better for some molecules like Chemotaxis and Cynovirin-N where much better accuracy is achieved with this information. For others too, the lack in accuracy is made up for by the save in time. As is evident, in each case the red line is always lower than the blue, indicating that similar (or better) RMSD is achieved in lesser time.

5.5 Comparison with Known Hinges

Flexibility of protein molecules is critical to understanding their conformational change. Hinge motions are critical for molecular identification, and knowledge of their location can guide the sampling of protein conformations for docking. To further validate the work done here, we identified the hinge residues for these molecules where such information was available and measured the correlation with

Table 5.2: RMSD Results with limited search space.

Molecule	Path	RMSD†	Time(min)†
Calmodulin	$1CLL \longrightarrow 1CTR$	3.2Å	≈ 25
		2.8Å	≈ 40
	$2F3Y \longrightarrow 1CFD$	2.98Å	42
AdK	$1AKE \longrightarrow 4AKE$	2.2Å	≈ 25
		2.2Å	70
GroEL	$1SS8 \longrightarrow 1SX4$	3.1Å	≈ 35
		2.96Å	52
CheY	$3CHY \longrightarrow 1CHN$	4.2Å	≈ 55
		3.7Å	70
		3.2Å	150
Cyanovirin	$2EZM \longrightarrow 1L5B$	4.9Å	52
		3.5Å	92
		2.98Å	120
		2.98Å	135
RBP	$1URP \longrightarrow 2DRI$	2.5Å	100
		1.98Å	≈ 130

† The top number is the RMSD when a threshold was set. The bottom number is when the simulation was allowed to run the maximum number of iterations.

our results. Combined results of studies from [7, 106, 107, 108] are presented in Table 5.3. Each of these works identifies regions within the molecules that are flexible. The list of residues to perturb for inducing conformational changes in each of these molecules was compared with the residue set in the second column of Table-5.3. In most cases, the residues in the flexible region form over 15% of the list of contacting residues, if not more. However, there is an exception with Chemotaxis. Here, according to [106], which has 3CHY as one of the molecules it analyses, this protein is thermally quite stable and undergoes changes only around temperatures of 400K, wherein residue 76 is the most active one. The residue set 70-80 did not appear in the list we used for perturbation (although 67, 82, 84 did appear). The residue sets by the META-PSICOV algorithm are merely the ones that evolve together. They may or may not have an effect on the flexibility. The implementation in this work can be readily extended to any external information about flexible residues. This is the subject of current work.

It must be emphasized that the contacting-residue bias does not account for all of the points that undergo change in the conformational transition. It shows a grouping of buildup matches that have a higher likelihood of evolving and potentially represents the neighboring of hotspot residues. In future, we intend to combine this information with the rigidity analysis which will allude more credibility to the feasible paths. We also intend to use rigidity analysis in a way that reduces randomness and focuses on generating a single path closest to the actual

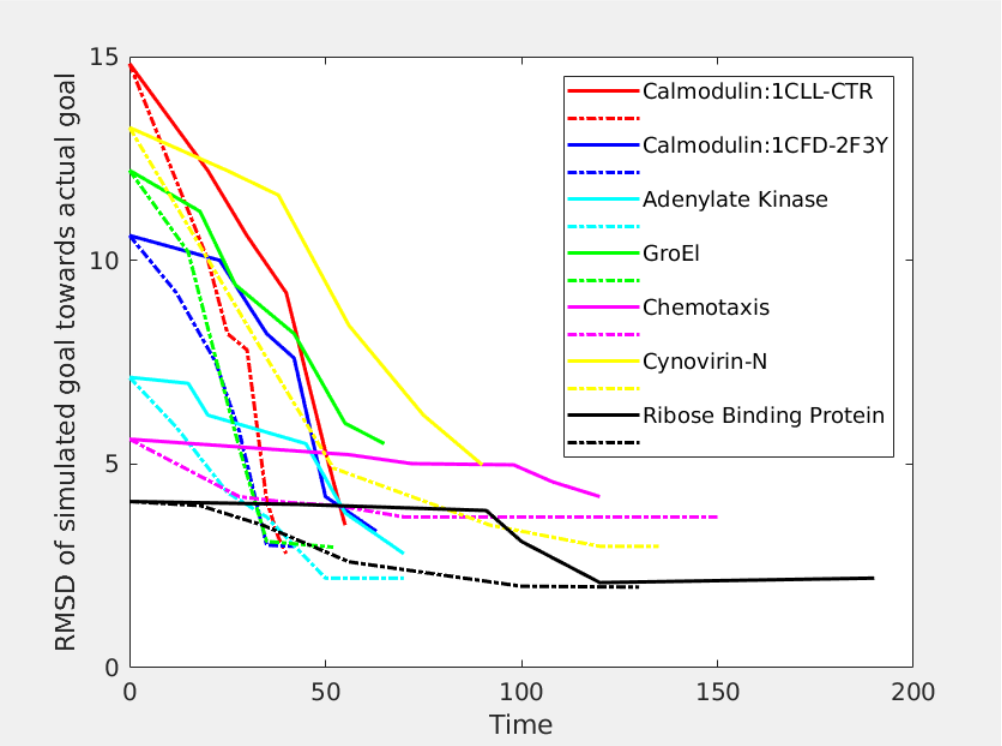


Figure 5.5: *RMSD towards goal versus time: the dotted lines show the simulation results coupled with contact-residue information and the solid lines without it*

Table 5.3: Hinge Residues of various molecules

Molecule	**Flexible Regions**
Calmodulin	73-82 [108]
Adenylate Kinase	113-124, 155-65, 174 [108, 7]
GroEl	16-32 (mobile loop), 19-376(apical domain)
Ribose Binding Protein	100-103, 234-237
Cynovirin-N	48-55
Chemotaxis	70-80 [106]

path that the molecules take.

Chapter 6

CONCLUSION

For the parallelized version of the feature reduction algorithm Isomap, besides the improvement in average time complexity due to betterment in computation of the shortest path tree, the algorithm offers better speed at every stage, when each function is compared in isolation. In future, we plan on attempting to do this even faster by combining the compute powers of the MPI and CUDA architectures described in the Introduction. The algorithm in its current form, only reduces the features of the data. Coming up with a way to reduce the noisy and redundant points themselves, is an area of ongoing research. As mentioned earlier, the Isomap algorithm is compute intensive. It executes in a way that hogs the system's resources, leaving all the other applications running on the system starved. In comparison, our parallelized version, efficiently makes use of the available memory, without having to write anything on the disk and hence doesn't hamper any other processes running on the system. The amount of data to be operated on is so large here, that programming platforms for vectorized data like Matlab and R sometimes cannot allocate memory for it. The coordinate data here is about 80 MB and the intermediate data files can be about 4GB. In situations like these, our version of the algorithm becomes the only resort.

The methods and results of Chapter-3 manifest the validation of our novel data instance reduction algorithm. The conclusions here are, however, far-reaching. The algorithm not only successfully reduces the data points required to represent a structure, it does so in a naturally cogent way. More complex the structure, lesser are the number of points reduced. In other words, the algorithm adheres to the complexities inherent in the data. The two dimensional projection of the swiss roll data set as expected is just a spiral and it is safe to assume that not too many points are required to assert that these points indeed represent a spiral. So, expectedly, the algorithm does away with the most number of points here. Amongst the macro-molecule datasets, Vasopressin (in these examples) is known

to be less structurally complex than Human Galanin and CDC42 and so, comparatively, ends up with a smaller portion of data points. A dataset representing the complexities of a visage intuitively has more intricacies to take care of and hence the Lena image data set ends up with a large portion of its original dataset. The Isolet dataset has 26 defined categories and so a training set for the best possible classification here eliminates lesser points than the Iris dataset.

The assessment of the information content offered by a data instance lets one decide whether to continue storing the particular data instance or not. The two Machine Learning datasets were used to draw conjectures of this sort. This could be an aid in determining and making sure that minimum number of data instances are stored. It would help in deciding whether new prospective data would affect the subsequent analysis or not. This is especially useful in large and complex data sets that require a lot of storage. If numerous attributes identify an instance, being able to do so is a boon. How this algorithm (or a variant of it) can be used for non-linear classification forms the basis for future work.

What lies ahead is to narrow down a way to obtain accurate projections of larger and relevant data sets using minimum possible data instances. A way to parameterize the dissimilarity of data instances would pave the way for non-linear feature reduction and then ultimately data reduction. One of our primary goals is to reduce protein data sets and use just enough points to isolate intermediate protein conformations [109, 110]. Their isolation would be key to characterization of their conformational landscape which would pave the way for understanding the convoluted relation between protein structure, dynamics and function.

Many proteins undergo large-scale conformational changes as part of their function. Characterizing the conformational space of proteins is crucial for understanding their function and dynamics. In Chapter-4 we presented an efficient filtration procedure that works well for sampling the intermediate conformations for protein molecules that undergo large scale conformational changes as well as for the ones that have a pervasive native state. The method presented is well suited to establish distinctiveness among the clusters generated. Refining these methods to produce finer sampling and study the significance of fleeting high energy conformations is the goal ahead. Hierarchical clustering merges (splits) two sets of conformations based on the heterogeneity of the entire set. In future, we aim at coming up with a way to bias this divide in a way that is more suited for protein datasets. Molecules that undergo rigorous changes in structure are the ones suited for such work. In particular, having known intermediates, would help in guiding the conformational pathway search problem as well. It can divide the search space into smaller in-

stances of the same problem. This is a portion of the ongoing research.

Chapter-5 lays the basis of actually simulating the actual physical pathways that proteins undertake. The proposed method effectively captures large-scale conformational changes and produces pathways that are consistent with experimental data and other computational studies. This work proposes a novel way to guide the conformational search process in finding a plausible path between two stable conformations of a protein molecule. The search is developed based on MCTS conformational sampling algorithm and it is biased by evolutionary information to guide the search and cut the run time. MCTS plays out a more comprehensive search than other stochastic based methodologies and because of the arbitrariness natural in the calculation and updating sampling iteratively, it yields various energetically doable ways between the conformations, which meet in view of the biasing towards the goal. The method represents an important first step towards a larger scale modeling of more complex biological systems. Future directions of this work include the integration of the coarse-grained and semi-coarse grained sampling into a multi-resolution search scheme, where the level of resolution can be determined on the fly according to the desired region in the conformational space to be sampled. The missing details can then be completed and interesting regions in the conformational space can be sampled in full representation. We also plan to scale up to larger proteins and protein complexes through multi-scale search and parallel computing.

BIIBLIOGRAPHY

[1] A. S. Tanenbaum and T. Austin, *Structured Computer Organisation*, 6th ed. Pearson, 2013.

[2] A. Grama, A. Gupta, G. Karypis, and V. Kumar, *Introduction to Parallel Computing*. Addison Wesley, 2003.

[3] C. Gibas and P. Jambeck, *Bioinformatics Computer Skills*. O'Reilly, 2001.

[4] K. Ocana and D. de Oliveira, "Parallel computing in genomic research: Advances and applications," 2015. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4655901/

[5] D. Case, T. Cheatham, T. Darden, H. Gohlke, R. Luo, K. M. Jr., A. Onufriev, C. Simmerling, B. Wang, and R. Woods, "The amber biomolecular simulation programs." *J. Computat. Chem.*, vol. 26, pp. 1668–1688, 2005.

[6] S. Kirkpatrick, C. D. G. Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[7] N. Haspel, M. Moll, M. Baker, W. Chiu, and L. E. Kavraki, "Tracing conformational changes in proteins," *BMC Structural Biology*, vol. Suppl1, p. S1, 2010.

[8] B. Raveh, A. Enosh, O. Furman-Schueler, and D. Halperin, "Rapid sampling of molecular motions with prior information constraints," *Plos Comp. Biol.*, vol. 5(2), p. e1000295, 2009.

[9] A. Shehu and B. Olson, "Guiding the search for native-like protein conformations with an ab-initio tree-based exploration," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1106–1127, 2010. [Online]. Available: http://ijr.sagepub.com/content/29/8/1106.abstract

[10] I. Al-Bluwi, M. Vaisset, T. Siméon, and J. Cortés, "Modeling protein conformational transitions by a combination of coarse-grained normal mode analysis and robotics-inspired methods," *BMC structural biology*, vol. 13, no. Suppl 1, p. S2, 2013.

[11] W. Zheng and B. Brooks, "Identification of dynamical correlations within the myosin motor domain by the normal mode analysis of an elastic network model," *J. Mol. Biol.*, vol. 346, no. 3, pp. 745–759, 2005.

[12] L. Yang, G. Song, and R. L. Jernigan, "Protein elastic network models and the ranges of cooperativity," *Proceedings of the National Academy of Sciences*, vol. 106, no. 30, pp. 12 347–12 352, 2009.

[13] G. Schroeder, A. T. Brunger, and M. Levitt, "Combining efficient conformational sampling with a deformable elastic network model facilitates structure refinement at low resolution," *Structure*, vol. 15, pp. 1630–1641, 2007.

[14] V. Frappier, M. Chartier, and R. J. Najmanovich, "Encom server: exploring protein conformational space and the effect of mutations on protein function and stability," *Nucleic Acids Research*, vol. 43, pp. W395–W400, 2015.

[15] D. Weiss and M. Levitt, "Can morphing methods predict intermediate structures?" *J. Mol. Biol.*, vol. 385, pp. 665–674, 2009.

[16] P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[17] L. Van Der Maaten, E. Postma, and J. Van den Herik, "Dimensionality reduction: a comparative review," *J Mach Learn Res*, vol. 10, pp. 66–71, 2009.

[18] J. Tenenbaum, V. de Silva, and J. Langford, "A global geometric framework for nonlinear dimensionality reduction." *Science*, pp. 2319–2323, 2000.

[19] P. Das, M. Moll, H. Stamati, L. Kavraki, and C. Clementi, "Low dimensional, free energy landscapes of protein folding reactions by nonlinear dimensionality reduction," *Proc. Nat. Acad. Sci.*, vol. 103, no. 26, pp. 9885–9890, 2006.

[20] A. Vajdi and N. Haspel, "A new dp algorithm for comparing gene expression data using geometric similarity," *IEEE International Conference on Bioinformatics and Biomedicine*, pp. 1157 – 1161, 2016.

[21] V. D. Silva and J. B. Tenenbaum, "Global versus local methods in nonlinear dimensionality reduction," *Advances in neural information processing systems*, 2003.

[22] T. Ameet, S. Kumar, and H. Rowley, "Large-scale manifold learning." *IEEE Conference on, Computer Vision and Pattern Recognition*, 2008.

[23] D. Luo, E. González, and N. Haspel, "Detecting intermediate protein conformations using algebraic topology," *BMC Bioinformatics*, vol. 18(Suppl 15), p. 502, 2017.

[24] Z. Yan and Q. Song, "An implementation of parallel floyd-warshall algorithm based on hybrid mpi and openmp," in *Proceedings of the 2012 International Conference on Electronics, Communications and Control*, ser. ICECC '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 2461–2466. [Online]. Available: https://doi.org/10.1109/ICECC.2012.339

[25] R. Sedgewick and K. Wayne, *Algorithms*, 4th ed. Addison-Wesley Professional, 2011.

[26] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Mag A*, pp. 59–572, 1901.

[27] M. Benito and D. Pena, "A fast approach for dimensionality reduction with image data." *Pattern recognition*, vol. 38, pp. 2400–2408, 2005.

[28] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20150202, 2016.

[29] E. J. Candes, X. Li, Y. Ma, and J. N. Wright, "Robust principal component analysis?" *ArXiv*, vol. abs/0912.3599, 2011.

[30] N. Locantore, J. Marron, D. Simpson, N. Tripoli, J. Zhang, and K. Cohen, "Robust principal component analysis for functional data," *Socicdad de Estadistica e Investigacion Operativa Test*, vol. 8, 1999.

[31] M. Greberman, "Image processing applications: An overview," 1984. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2578700/

[32] A. Bovick, "Handbook of image and video processing," *Academic Press New York*, 2000.

[33] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Machine Learning*, 2000.

[34] A. A. Gonzalez, J.-F. Diez-Pastor, J. J. Rodriguez, and C. G. Osorio, "Istance selection of linear complexity for big data," *Knowledge Based Systems*, 2016.

[35] S. Garcia, J. Derrac, J. Cano, and F. Herrera, "Prototype selection for nearest neighbor classification: Taxonomy and empirical study," *IEEE's Transactions on Pattern Analysis and Machine Intelligence*, pp. 417–435, 2012.

[36] I. Czarnowski and P. Jedrzejowicz, "Instance reduction approach to machine learning and multi-database mining," *Annales UMCS Informatica*, 2006.

[37] S.-H. Son and J.-Y. Kim, "Data reduction for instance-based learning using entropy-based partitioning," in *Computational Science and Its Applications - ICCSA 2006*, M. Gavrilova, O. Gervasi, V. Kumar, C. J. K. Tan, D. Taniar, A. Laganá, Y. Mun, and H. Choo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 590–599.

[38] J. Fujiki and S. Akaho, "Spherical pca with euclideanization," *ACCV'07 Workshop Subspace*, November 2007.

[39] M. Fontes and C. Soneson, "The projection score - an evaluation criterion for variable subset selection in pca visualization," *BMC Bioinformatics*, vol. 12, no. 1, p. 307, Jul 2011. [Online]. Available: https://doi.org/10.1186/1471-2105-12-307

[40] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kale, and K. Schulten, "Scalable molecular dynamics with namd," *Journal of computational chemistry*, vol. 26, no. 16, pp. 1781–1802, 2005.

[41] K. Wennerberg, K. L. Rossman, and C. J. Der, "The ras superfamily at a glance," *Journal of Cell Science*, vol. 118, no. 5, pp. 843–846, 2005. [Online]. Available: http://jcs.biologists.org/content/118/5/843

[42] R. C. Gonzalez and R. E. Woods, *Digital image processing*. Upper Saddle River, N.J.: Prentice Hall, 2008. [Online]. Available: http://www.amazon.com/Digital-Image-Processing-3rd-Edition/dp/013168728X

[43] A. Joshi, A. Boyat, and B.K.Joshi, "Impact of wavelet transform and median filtering on removal of salt and pepper noise in digital images," *Proc. of International Conference on Issues and Challenges in Intelligent Computing Techniques*, 2014.

[44] A. H. Ashtari, "Pic/plot images to coordinate points," National University of Malaysia, Center for Artificial Intelligence Technology (CAIT), Faculty Of Information Science and Technology, 2015.

[45] P.-A. Cazade, W. Zheng, D. Prada-Gracia, G. Berezovska, F. Rao, C. Clementi, and M. Meuwly, "A comparative analysis of clustering algorithms: O2 migration in truncated hemoglobin i from transition networks," *The Journal of Chemical Physics*, vol. 142, no. 2, pp. –, 2015. [Online]. Available: http://scitation.aip.org/content/aip/journal/jcp/142/2/10.1063/1.4904431

[46] R. Fisher, "The use of multiple measurements in taxonomic problems," *Annual Eugenics 7, Part II*, 1936.

[47] B. Dasarathy, "Nosing around the neighborhood: A new system structure and classification rule for recognition in partially exposed environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, No. 1, 67-71,*, 1980.

[48] D. Simovici, *Mathematical Analysis for Machine Learning and Data Mining*. World Scientific, 2018.

[49] M. A. Fanty and R. Cole, "Spoken letter recognition," in *NIPS*, 1990, p. 220.

[50] T. G. Dietterich and G. Bakiri, "Error-correcting output codes: A general method for improving multiclass inductive learning programs," in *AAAI*, 1991, p. 572.

[51] ——, "Solving multiclass learning problems via errorcorrecting codes," *Journal of Artificial Intelligence Research*, vol. 2:, pp. 263–268, Jan 1995.

[52] M. Karplus and E. Shakhnovitch, "Protein folding: Theoretical studies of thermodynamics and dynamics." *Creighton t, ed. edition*, p. 127–195, 1992.

[53] J. D. Bryngelson, J. N. Onuchic, N. D. Socci, and P. G. Wolynes, "Funnels, pathways, and the energy landscape of protein folding:a synthesis." *Proteins*, vol. 21, p. 167–195, 1995.

[54] A. Joshi and N. Haspel, "Principal component analysis based dimensionality reduction technique," *submitted to Springer's, Machine Learning*, 2019.

[55] ——, "Clustering of protein conformations using parallelized dimensionality reduction," *Journal of Advances in Information Technology*, 2019.

[56] E. Pettersen, T. Goddard, C. Huang, G. Couch, D. Greenblatt, E. Meng, and T. Ferrin, "Ucsf chimera–a visualization system for exploratory research and analysis," *J. Comput. Chem.*, vol. 25, no. 13, pp. 1605–1612, 2004.

[57] W. L. Jorgensen, J. Chandrasekhar, J. Madura, R. W. Impey, and M. L. Klein, "Comparison of simple potential functions for simulating liquid water," *J. Chem. Phys.*, vol. 79, pp. 926–935, 07 1983.

[58] T. Darden, D. York, and L. Pedersen, "Particle mesh ewald: An n*log(n) method for ewald sums in large systems." *J. Chem. Phys.*, vol. 98(12), p. 10089–10092, 1993.

[59] L. Kale, R. Skeel, M. Bhandarkar, R. Brunner, A. Gursoy, N. Krawetz, J. Phillips, A. Shinozaki, K. Varadarajan, and K. Schulten, "Namd2: Greater scalability for parallel molecular dynamics." *J Comp. Phys.*, vol. 151, p. 283–312, 1999.

[60] Y. Duan, C. Wu, S. Chowdhury, M. C. Lee, G. Xiong, W. Zhang, R. Yang, P. Cieplak, R. Luo, T. Lee, J. Caldwell, J. Wang, and P. Kollman, "A point-charge force field for molecular mechanics simulations of proteins based on condensed-phase quantum mechanical calculations," *Journal of Computational Chemistry*, vol. 24, no. 16, pp. 1999–2012, 2003. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.10349

[61] D. Luo and N. Haspel, "Multi-resolution rigidity-based sampling of protein conformational paths," in *CSBW (Computational Structural Bioinformatics Workshop), in proc. of ACM-BCB (ACM International conference on Bioinformatics and Computational Biology)*, September 2013, pp. 787–793.

[62] Z. Cang, E. Munch, and G.-W. Wei, "Evolutionary homology on coupled dynamical systems," *arXiv preprint arXiv:1802.04677*, 2018.

[63] P. G. Cámara, "Topological methods for genomics: present and future directions." *Current opinion in systems biology, vol. 1*, pp. 95–101, 2017.

[64] G. Wei-Wei, "Persistent homology analysis of biomolecular data," *Society for Industrial and Applied Mathematics*, 2017. [Online]. Available: https://sinews.siam.org/Details-Page/persistent-homology-analysis-of-biomolecular-data

[65] H.-W. Chang, S. Bacallado, V. S. Pande, and G. E. Carlsson, "Persistent topology and metastable state in conformational dynamics," *PLoS ONE*, vol. 8, no. 4, p. e58699, 04 2013. [Online]. Available: http://dx.doi.org/10.1371%2Fjournal.pone.0058699

[66] H. Adams and A. Tausz, "Javaplex: A research software package for persistent (co)homology," Software available at http://code.google.com/javaplex, Stanford University, 2011.

[67] A. Tausz, M. Vejdemo-Johansson, and H. Adams, "JavaPlex: A research software package for persistent (co)homology," in *Proceedings of ICMS 2014*, ser. Lecture Notes in Computer Science 8592, H. Hong and C. Yap, Eds., 2014, pp. 129–136, software available at http://appliedtopology.github.io/javaplex/.

[68] R. R. Wadhwa, D. F. Williamson, A. Dhawan, and J. G. Scott, "Introduction to persistent homology with tdastats," *The Journal of Open Sorce Software*, 2018. [Online]. Available: https://cran.r-project.org/web/packages/TDAstats/vignettes/intro.html

[69] F. Valdés-Mora, T. G. del Pulgar, and J. C. Lacal, Translational Oncology Unit CSIC-UAM- La Paz Centro Nacional de Biotecnologia C/ Darwin 3, Campus de Cantoblanco, 28049 Madrid, Spain, 2012.

[70] M. Maldonado and S. Dharmawardhane, "Targeting rac and cdc42 gtpases in cancer." 2018.

[71] H. Caldwell and W. I. Young, "Oxytocin and vasopressin: Genetics and behavioral implications," in *Handbook of Neurochemistry and Molecular Neurobiology: Neuroactive Proteins and Peptides*, 3rd ed. Berlin, Germany: Springer, 2006, pp. 573–607.

[72] H. Evans, M. Baumgartner, J. Shine, and H. Herzog, "Genomic organization and localization of the gene encoding human preprogalanin," *Genomics*, vol. 18(3), pp. 473–477, 1993.

[73] M. Bersani, A. H. Johnsen, P. Hojrup, B. E. Dunning, J. J. Andreasen, and J. J. Holst, "Human galanin: primary structure and identification of two molecular forms," *Federation of European Biochemical Sciences*, vol. 283, pp. 189–194, 1991.

[74] Y. Feng, L. Yang, A. Kloczkowski, and R. Jernigan, "The energy profiles of atomic conformational transition intermediates of adenylate kinase," *Proteins*, vol. 77(3), pp. 551–558, 2009.

[75] H. Jónsson, G. Mills, and K. W. Jacobsen, "Nudged elastic band method for finding minimum energy paths of transitions," in *CLassical and Quantum Dynamics in Condensed Phase Simulations*, B. J. Berne, G. Ciccotti, and D. F. Coker, Eds., Jun. 1998, pp. 385–404.

[76] M. S. Apaydın, D. L. Brutlag, C. Guestrin, D. Hsu, and J.-C. Latombe, *Stochastic Conformational Roadmaps for Computing Ensemble Properties of Molecular Motion.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 131–147. [Online]. Available: https://doi.org/10.1007/978-3-540-45058-0_9

[77] D. Guieysse, J. Cortés, M. Remaud-Siméon, T. Siméon, V. Ruiz de Angulo, and V. Tran, "A path planning approach for computing large-amplitude motions of flexible molecules," *Bioinformatics*, vol. 21, pp. 116–125, 06 2005. [Online]. Available: https://dx.doi.org/10.1093/bioinformatics/bti1017

[78] K. Molloy and A. Shehu, "Elucidating the ensemble of functionally-relevant transitions in protein systems with a robotics-inspired method," *BMC Struct. Biol.*, vol. 13, no. Suppl 1, p. S8, 2013.

[79] J. B. Brokaw and J.-W. Chu, "On the roles of substrate binding and hinge unfolding in conformational changes of adenylate kinase," *Biophysical Journal*, vol. 99, 2010.

[80] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002. [Online]. Available: https://doi.org/10.1177/027836402320556421

[81] A. M. Ladd and L. E. Kavraki, "Motion planning in the presence of drift, underactuation and discrete system changes," in *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2005.

[82] S. M. LaValle and J. James J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001. [Online]. Available: https://doi.org/10.1177/02783640122067453

[83] K. P. Molloy, "Probabilistic algorithms for modeling protein structure and dynamics," Ph.D. dissertation, George Mason University, 2015.

[84] N. M. Amato, "Sampling-based motion planning: From intelligent cad to crowd simulation to protein folding (invited talk)," in *16th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2018)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), D. Eppstein, Ed., vol. 101. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, pp. 1:1–1:1. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2018/8827

[85] E. Hruska, J. R. Abella, F. Nüske, L. E. Kavraki, and C. Clementi, "Quantitative comparison of adaptive sampling methods for protein dynamics," *The Journal of Chemical Physics*, vol. 149, no. 24, p. 244119, 2018. [Online]. Available: https://doi.org/10.1063/1.5053582

[86] N. Haspel, D. Luo, and E. González, "Detecting intermediate protein conformations using algebraic topology," *BMC bioinformatics*, vol. 18, no. 15, p. 502, 2017.

[87] K. Molloy and A. Shehu, "A general, adaptive, roadmap-based algorithm for protein motion computation," *IEEE Transactions on NanoBioscience*, vol. 15, pp. 158–165, 2016.

[88] A. B. Zaman and A. Shehu, "Balancing multiple objectives in conformation sampling to control decoy diversity in template-free protein structure prediction," *BMC Bioinformatics*, vol. 20, no. 1, p. 211, Apr 2019. [Online]. Available: https://doi.org/10.1186/s12859-019-2794-5

[89] A. Estaña, K. Molloy, M. Vaisset, N. Sibille, T. Siméon, P. Bernadó, and J. Cortés, "Hybrid parallelization of a multi-tree path search algorithm: Application to highly-flexible biomolecules," *Parallel Computing*, vol. 77, pp. 84 – 100, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167819118301893

[90] J. Chen, J. Wang, and W. Zhu, "Zinc ion-induced conformational changes in new delphi metallo-β-lactamase 1 probed by molecular dynamics simulations and umbrella sampling," *Physical Chemistry Chemical Physics*, vol. 19, no. 4, pp. 3067–3075, 2017.

[91] Z. Zhang, U. Ehmann, and M. Zacharias, "Monte carlo replica-exchange based ensemble docking of protein conformations," *Proteins: Structure, Function, and Bioinformatics*, vol. 85, no. 5, pp. 924–937, 2017.

[92] G. E. Karagöz, D. Acosta-Alvear, H. T. Nguyen, C. P. Lee, F. Chu, and P. Walter, "An unfolded protein-induced conformational switch activates mammalian ire1," *Elife*, vol. 6, p. e30700, 2017.

[93] L. Chon, A. S. Saglam, and D. M. Zuckerman, "Path-sampling strategies for simulating rare events in biomolecular systems," *Elsevier's Current Opinion in Structural Biology*, vol. 43, pp. 88–94, 2017.

[94] C. Ekenna, S. Thomas, and N. M. Amato, "Adaptive local learning in sampling based motion planning for protein folding," *BMC systems biology*, vol. 10, no. 2, p. 49, 2016.

[95] M. Schelling, T. A. Hopf, and B. Rost, "Evolutionary couplings and sequence variation effect predict protein binding sites," *Proteins: Structure, Function, and Bioinformatics*, vol. 86, no. 10, pp. 1064–1074, 2018.

[96] D. T. Jones, D. W. A. Buchan, D. Cozzetto, and M. Pontil, "PSICOV: precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments," *Bioinformatics*, vol. 28, no. 2, pp. 184–190, 11 2011. [Online]. Available: https://doi.org/10.1093/bioinformatics/btr638

[97] S. Ovchinnikov, H. Kamisetty, and D. Baker, "Robust and accurate prediction of residue–residue interactions across protein interfaces using evolutionary information," *Elife*, vol. 3, p. e02030, 2014.

[98] A. V. Hoojghan, "Application of graphical models in protein-protein interactions and dynamics," Ph.D. dissertation, University of Massachusetts Boston, 2018.

[99] O. Keskin, A. Gursoy, B. Ma, and R. Nussinov, "Principles of protein- protein interactions: What are the preferred ways for proteins to interact?" *Chemical reviews*, vol. 108, no. 4, pp. 1225–1244, 2008.

[100] G. A. Papoian, J. Ulander, M. P. Eastwood, Z. Luthey-Schulten, and P. G. Wolynes, "Water in protein structure prediction," *Proceedings of the National Academy of Sciences*, vol. 101, no. 10, pp. 3352–3357, 2004.

[101] Y. Duan, C. Wu, S. Chowdhury, M. C. Lee, G. Xiong, W. Zhang, R. Yang, P. Cieplak, R. Luo, T. Lee *et al.*, "A point-charge force field for molecular mechanics simulations of proteins based on condensed-phase quantum mechanical calculations," *Journal of computational chemistry*, vol. 24, no. 16, pp. 1999–2012, 2003.

[102] A. Vajdi and N. Haspel, "Clustering protein conformations using a dynamic programming based similarity measurement," *BICOB*, 2016.

[103] D. T. Jones, T. Singh, T. Kosciolek, and S. Tetchner, "MetaPSICOV: combining coevolution methods for accurate prediction of contacts and long range hydrogen bonding in proteins," *Bioinformatics*, vol. 31, no. 7, pp. 999–1006, 11 2014. [Online]. Available: https://doi.org/10.1093/bioinformatics/btu791

[104] B. AJ and M. SL, "Multiple open forms of ribose-binding protein trace the path of its conformational change." *Journal of Molecular Biology*, vol. 279, 1998.

[105] M. J. Cuneo, L. S. Beese, and H. W. Hellinga, "Ligand-induced conformational changes in a thermophilic ribose-binding protein," *BMC Structural Biology*, vol. 8, no. 1, p. 50, Nov 2008. [Online]. Available: https://doi.org/10.1186/1472-6807-8-50

[106] Y. He, G. G. Maisuradze, Y. Yin, K. Kachlishvili, S. Rackovsky, and H. A. Scheraga, "Sequence-, structure-, and dynamics-based comparisons of structurally homologous chey-like proteins," *Proceedings of the National Academy of Sciences*, vol. 114, no. 7, pp. 1578–1583, 2017. [Online]. Available: https://www.pnas.org/content/114/7/1578

[107] J. Ma and M. Karplus, "The allosteric mechanism of the chaperonin groel: A dynamic analysis," *Proceedings of the National Academy of Sciences*, vol. 95, no. 15, pp. 8502–8507, 1998. [Online]. Available: https://www.pnas.org/content/95/15/8502

[108] K. Keating S, S. Flores C, M. Gernstein B, and L. Kuhn A., "Stonehinge: Hinge prediction by network analysis of individual protein structures," *Proyein Science*, vol. 18, pp. 359–371, 2008.

[109] S. Kirkpatrick, C. D. G. Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[110] R. Vetro, N. Haspel, and D. Simovici, "Characterizing intermediate conformations in protein conformational space," in *Proc. of the Ninth International Meeting on Computational Intelligence Methods for Bioinformatics and Biostatistics*, Houston, TX, USA, July 2012.