Project Report: Indexing Live Sequences Over a Large Timescale

Friday, December 8, 2023

Tad Berkery, Lily Chen, Richard Hu, Yuseong “Nick” Oh

Abstract

In this paper, we present a novel approach to analyzing Molecular Dynamics (MD) simulations, which are pivotal in understanding the complex conformational changes of proteins over extended timescales. Despite the effectiveness of MD simulations, they are notably memory-intensive, capturing detailed protein movements at minuscule intervals. To address this, we employ the Markov State Model to efficiently map the vast array of potential protein conformations into approximate states, allowing us to track the transitions between these states over time without storing exhaustive coordinate data at every timestep. Our exploration includes the use of BWT-HMM (Burrows-Wheeler Transform – Hidden Markov Model) and the Viterbi algorithm for compressing these time-dependent, probabilistic, and highly variable sequences with high repetition rates. Moreover, we introduce an innovative method to merge BWT-HMM models, achieving greater space efficiency than traditional approaches. This work not only offers a more efficient way to manage protein simulation data, but also aids in identifying common sequence motifs within protein conformations potentially unlocking new insights into their physiological significance.

Introduction

Computational protein simulations represent a cornerstone in modern bioinformatics and biophysical research, offering profound insights into the intricate mechanisms of proteins at an atomic level [1]. These simulations utilize computational methods to model the behavior of proteins in various environments, elucidating their structure, dynamics, and interactions. The applications of computational protein simulations are vast, ranging from understanding disease mechanisms and protein folding pathways to aiding in the design of novel drugs. These simulations leverage the power of algorithms and high-performance computing to predict protein behavior. This predictive capability is crucial in fields like drug discovery, helping the identification of drug targets and protein interactions, allowing researchers to test hypotheses about protein function and dynamics that are not easily accessible via experimental methodologies [1].

Molecular Dynamics (MD) simulations, a specific subset of computational protein simulations, are particularly noteworthy for their ability to provide time-resolved information about protein movements and interactions. MD simulations offer a dynamic view of proteins, simulating their behavior over time under various physiological conditions. This method calculates the trajectories of atoms in a protein by solving Newton's equations of motion, thereby providing insights into the conformational changes, folding mechanisms, and interaction dynamics of proteins [2]. The strength of MD simulations lies in their ability to model protein behavior in environments closely mimicking biological systems, including the presence of solvents, ions, and other macromolecules. This detailed representation is invaluable for understanding the molecular basis of enzyme catalysis, protein-ligand interactions, and the effect of mutations on protein stability and function. MD simulations have been instrumental in areas like drug design, where they help in identifying the binding modes of small molecules and in predicting the effects of mutations on drug resistance [2].

By enabling detailed analysis of protein structures and functions at the molecular level, these simulations have revolutionized our approach to understanding life's fundamental processes. A critical advantage of protein simulations is their ability to explore vast conformational spaces, providing insights into protein dynamics that are often elusive in experimental setups due to time or technical constraints. This is particularly crucial in understanding phenomena like protein folding, misfolding, and complex formation, which are fundamental to cellular functions and various diseases. Protein simulations have also been instrumental in the field of drug discovery and development, allowing for the *in silico* screening of compounds, predicting drug efficacy, and understanding resistance mechanisms [1].

A significant disadvantage of MD simulations lies in their extensive demand for computing resources and the time required for computation. These simulations, particularly high-resolution and long-duration studies, necessitate vast computational power. This high demand translates into substantial costs and logistical challenges in conducting and maintaining such research. Furthermore, most simulations are still limited to modeling protein behaviors over relatively short timescales, typically in the order of nanoseconds to microseconds. This limitation constrains our ability to fully understand longer-term biological processes and events that occur over milliseconds or seconds, potentially leaving critical aspects of protein dynamics and interactions unexplored. The computational intensity and time constraints thus impose significant limitations on the scope and depth of protein simulation studies.

One way to simplify the protein conformations that results from the MD simulation is approximating with the Markov State Model (MSM). In MSM, the conformation of the protein at each step is assigned to a discrete state (e.g. A, B, or C). The result is a sequence of protein states (e.g. AAABBC). The MSM then calculates the probability of transitioning from one state to any given state, which is described in the form of a transition matrix [3]. One key assumption of the MSM model is that each state only depends on the preceding state [3], which often may not be true in many physiological contexts. Our goal is to combine the MSM with other data structures to store and query the conformational sequence more efficiently while capturing long range sequence motifs.

In addressing the significant challenges posed by the memory and computational demands of protein simulations, our project pivoted towards exploring innovative data structure implementations. The Burrows-Wheeler Transform with Hidden Markov Models (BWT-HMM) emerged as a promising method to potentially alleviate the memory constraints inherent in traditional approaches. The rationale behind selecting BWT-HMM for our investigation stems from its proven efficacy in other data-intensive domains, like genomics, where it has demonstrated a capacity to efficiently handle and compress large datasets. By applying BWT-HMM to protein simulations, we aimed to optimize data management, reduce memory requirements, and enhance the computational efficiency of these simulations. Our project was driven by the objective of unlocking new potentials in protein simulations, making them more accessible and feasible, especially for long-term and large-scale studies.

Knowing common sequence motifs may reveal important dynamical processes of the protein such as the folding pathway or mechanism of enzyme catalysis. Although a k-mer index is a good starting point for identifying frequently occurring subsequences, it is very prone to noise and may identify slightly modified versions of the same sequence motif. The Viterbi algorithm presents an effective way to identify sequence motifs within the protein conformational sequence. The Viterbi algorithm is a dynamic programming algorithm for finding the most probable path. By applying the Viterbi algorithm to protein simulations, we aim to identify common sequence motifs within the protein conformational sequence by treating it as a most probable path problem.

Prior Work

Part 1: Compress the Live Sequence

Our project's exploration into the use of BWT-HMM for protein simulations is grounded in significant prior research in the field. The Burrows-Wheeler Transform (BWT) first gained attention in bioinformatics for its data compression capabilities, particularly through Langmead et al.'s development of the Bowtie algorithm [4]. This work illustrated how BWT could effectively reduce memory requirements for genomic sequence alignment, suggesting its broader applicability in managing large biological datasets.

In parallel, Hidden Markov Models (HMMs) became essential tools in computational biology for sequence analysis, especially rooted from Eddy's work on protein family modeling [5]. HMMs proved effective in modeling biological sequences, providing a valuable complement to the data management capabilities of BWT.

However, the combined application of BWT and HMMs has been less explored, particularly in protein simulations. A 2020 study [6] suggests an initial investigation into this combination, showing promise for efficient sequence analysis in large genomic databases. Our project builds on these foundations, aiming to leverage BWT's data compression strength and HMM's sequence modeling accuracy to address the challenges of computational and memory efficiency in protein simulations.

Part 2: Find Sequence Motifs

After much consideration, we proposed that traditional algorithms to find sequence motifs may not be the best “prior work” to inspire a new method. Foremost, consider the unit signal we are reading:

> Genomics: The base pairs within relatively fixed indexes of the individual’s genome, encoded as {A, C, G, T}. While mutations, insertions, deletions, and structural variations are a challenge to the field, matching/local alignment algorithms can give you a reasonable estimate of what the short repeat motifs are, how frequent they occur, and where they are in the genome.
>
> MSM Molecular Dynamics: Again, our data is now: (1) **time-varying,** (2) much more **dependent on previous states,** and (3) **highly stochastic** compared to “reading the next DNA base pair in a stable human genome.” We are using a very small time interval to sample how a single protein is thermodynamically fluctuating (within equilibrium of its folded state), as we won’t know exactly when our protein will transition from one conformation to the next.

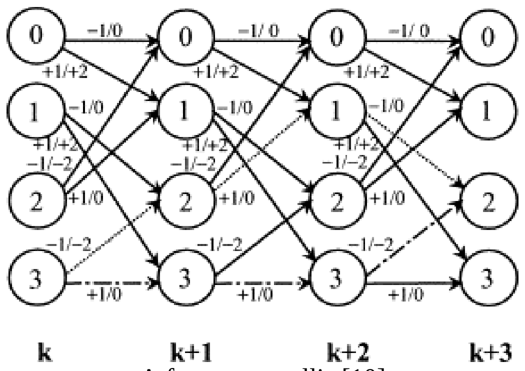
Most of our sequence data ended up resembling “*AAAAAA...AAABBBB...BBBCCCCC...CCCCCBBBB...”* which means that the most frequently occurring repeats are single long runs. Could we simply compress single-character runs into a compact representation, then run a repeat finding algorithm to look for *k*-length patterns? This might be a valid approach if the protein has highly deterministic folding paths, but in MSM molecular dynamics, this is often not the case. Although the ground-truth motif might be “ABCD,” our simulation will have all kinds of mutations—mismatches (“A**D**CD”), insertions (“AB**EB**CD”), and deletions (“ABC”)—as a result of random forces in our standard biophysical model (see: Langevin dynamics). Over the first few weeks, we also came up with a long list of additional, small reasons as to why repeat finding/local alignment might not work very well for our problem. Two more highlights:

1. The full sequence motif will occur with Markovian probability: P(ABCDEFG) = P(A) P(B|A) ... P (G|F). Small sub-patterns might occur frequently (ABC, BCD, ...), but to see most of an entire subsequence of even k>5, each conditional probability has to be very high.
2. MSM molecular dynamics simulations often use a large alphabet (e.g.: 55), which can increase noise. [7]

Instead, we propose that a better way to predict the ground truth is to aggregate sequence evidence from across the entire simulation, using them to extract some consensus signal that masks the high individual variance. To do this, we draw on a problem/algorithm from an adjacent field, which also applies a concept from the class.

<u>Inspiration/prior work:</u> In signal processing, Maximum Likelihood Sequence Estimation describes the challenge of extracting the most important or highest-probability signal—a sequence of inputs— from a

channel that contains high inter-symbol interference or noise. The Viterbi algorithm was indeed first developed for this field, although later variations such as HMM prediction followed the first publication in 1967 [8]. A few years later, G. David Forney conceptualized the algorithm as a trellis, which graphically summarizes all possible sequences that were observed during a recording [9]. We show a more recent representation of the trellis below:



A four-state trellis [10]

In this design, the most likely signal can be found as the heaviest path traversing from layer *k* to layer *k*+3, although the capacities of edges are defined differently due to field-specific considerations. [10]

We further modified Viterbi to be compatible with our problem space of interest, as described in future sections. We also implemented some enhancements to Viterbi developed in the past 50 years:

> Merging algorithm to reduce the complexity of the trellis: Maximum Likelihood Sequence Estimation aims to detect a strong signal that can begin at any timestep in the full recording. To graphically encode this, the complexity would be exponential with respect to the number of nodes, and multiplicatively worse for parsing the graph. A 2019 IEEE conference paper proposed an algorithm to merge nodes with identical incoming or outgoing edges, reducing complexity without notably reducing performance. To visualize this problem, we would especially recommend Figures 2-6. Our problem does not require that we implement the same algorithm, but we similarly take advantage of reducing the complexity of our graph. [11]
>
> Beam search with dynamic pruning: Beam search aims to reduce the complexity of the problem by only considering a fixed number of high-capacity edges. We similarly removed edges with significantly low capacities without affecting the results of our simulations. [12]
>
> Dealing with insertions and deletions: We found “Problem 2” described in Schwartz et al. to especially inform one of our core challenges: handling noisy insertions and deletions in the data stream. Due to our testing and time constraints, we tested a variation of Figure 5c. Another interesting consideration was different implementations of optimum path, but this did not notably affect our results. [13]

Methods and Software

Part 1: BWT-HMM

In approaching our objectives, we developed a method that synergizes the Burrows-Wheeler Transform (BWT) with Hidden Markov Models (HMM) for enhanced analysis of protein conformations. Utilizing the hmmlearn library in Python, we applied BWT as a preprocessing step to reorganize the numerical protein conformation sequences, which improves the efficiency of the HMM's analysis of state transitions. The transformed sequence, clustered by similar states, provides a refined input for the HMM, which is then modeled as a discrete Multinomial process. Given the discrete nature of protein conformations, represented numerically from 1 through 6, this approach is well-suited for capturing the dynamics of protein folding and interactions.

The HMM was then initialized with uniform probabilities across its states and transitions, reflecting an unbiased starting point for the learning process. The Expectation-Maximization algorithm employed by hmmlearn iteratively adjusts these probabilities to fit the data, resulting in a model that encapsulates the statistical properties of protein state transitions. After training, the model can simulate new protein conformation sequences and predict transitions, offering valuable insights into the protein folding process. This dual BWT-HMM method presents a compact, data-driven approach for handling the complexity of protein simulations, promising to improve computational efficiency in large-scale biomolecular studies.

To optimize our BWT-HMM framework for protein conformation analyses, we implemented an innovative strategy that we labeled "Unique Conformation Compression." This technique is predicated on the observation that protein simulations frequently display sequences where specific conformations are repeated in succession. Our compression method identified and consolidated consecutive repetitions, which significantly reduced the sequence length, streamlining the subsequent BWT and HMM processing. Our rationale was twofold: this method reduces the computational load by minimizing redundant data and enhances the predictive performance of the model by focusing on the transitions between unique conformations. As such, this model is especially useful in predicting the next conformational state, as it allows the model to learn from a distilled representation of the sequence with reduced noise.

To further enhance the scalability and effectiveness of our BWT-HMM, we implemented a novel "BWT-HMM Merging" method. By segmenting the protein conformation sequence into smaller chunks and applying BWT to each segment, we created multiple compressed representations that retain the essential transition information. These segmented BWT outputs were then collectively analyzed using a Multinomial HMM to model the protein dynamics. This segmentation of the sequence addresses several issues in processing sequences. Firstly, we can generate BWTs in real-time, as "chunks" of sequences can be processed in parallel to the MD simulations. Further, the method aids bringing similar conformations closer, making it easier for the HMM to learn transition dynamics.

By implementing the Unique Conformation Compression and Merging methods, we enable our BWT-HMM model to discern and predict the most relevant conformation changes, which is the critical aspect of understanding protein molecular dynamics more effectively. Overall, these modifications allow for a more management and efficient handling of the conformational data, optimizing memory usage and improving the model's ability to learn from the sequence data for conformation predictions.

Part 2: Viterbi to Find Sequence Motifs

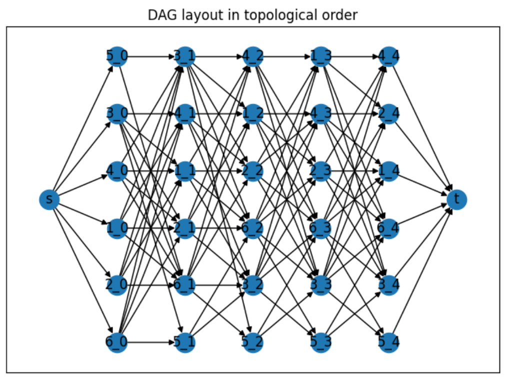
Modification 0: Adapt Viterbi for our motif search problem

We based our Viterbi algorithm on the trellis problem advanced by both Viterbi and Forney. In the standard approach, the goal of Viterbi is to provide the most likely "label" at every state of the measurement/simulation, provided as a single consensus sequence. In our modification, we are trying to adapt the principle of Viterbi to instead find common *k*-length motifs across the entire sequence.

In our initial proposal, a sliding window of specified length *k* is moved across the entire sequence to obtain sequences used for the trellis. First, you start with the first column in the trellis where the node matches the first state int the sliding window. After proceeding to the next state in the sliding window, you would add 1 to the path taken to go from state one in the first column to state 2 in the second column. This was continued for all states in the sliding window until the right most column is reached.

In the completed trellis, the heaviest path represents the mostly probable/consensus sequence. We used a dynamical programming approach to find the heaviest path. First, we start with the node that has the heaviest outgoing edge and walk to the next node via the heaviest path. We continue finding the heaviest path until we reach a node in the last column. Once the first path is found, the edges associated with the path are removed. This process is repeated until we find a designated number of “top n” motifs.

For the heaviest path, we also tested converting each edge from flow (raw number of sequences going between two nodes) into the –log(probability), just as we learned in class. This makes the heaviest path a Markovian consideration, but it didn’t end up notably changing our proposals. We kept this modification, which can be found in our dynamic programming calculations in main_method/propose_paths.py:build_dist_graph().

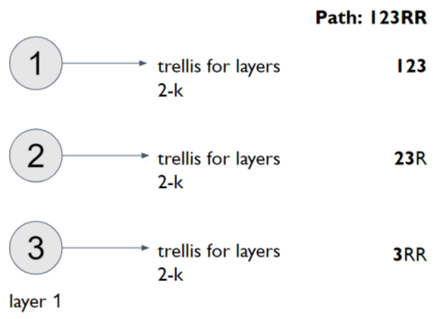


Simple visualization of our initial trellis with networkx.

Further Modification 1: Minimizing signal overlaps between the sliding window

One problem with the sliding window approach is that the most probable sequences output by the Viterbi algorithm contain random fragments surrounding the actual sequence. For example, given a sequence of XXXX12345XXXXX, sliding window would capture 12345 as a signal as well as various fragments surrounding the actual sequence. A sliding window of length 5 would capture 12345, 2345X, 345XX, and more. In addition, if there is significant noise, or if multiple sequence motifs share a common substring, our consensus signal may not be as robust now that we are using a sliding window.

To deal with these issues, we attempted to separate our trellis problem into subgraphs (shown below). For example, given a window of “EFG”, we would add flow to subgraph “E”, and we would add flow for window “FG” to subgraph “F.” This approach allows us to reduce graph overlap in all scenarios, but a subgraph will still carry a fraction of residual noise if the sequence motif has repeat characters (e.g. EFGE).



Modified problem

More importantly, separating our trellis problem into subgraphs could take advantage of the Markovian probability. To illustrate this problem, given a ground truth of ABCDEFG, the Markovian nature of our model makes it multiplicatively less likely that we see ABC, ABCD, …, and ABCDEFG as a consensus sequence. However, at each subgraph, we might strongly see A-->BCDEFG, B->CDEFG, …, F->G. We experimentally confirmed that our heaviest path problem extracts the above trends. It should be very feasible to use a variation of local alignment and best-buddy to piece ABCDEFG together from the various subgraphs. We also adapted other smaller modifications in our model, labeled "Further Modifications."

Further Modification 2: We confirmed that removing low-capacity edges did not change the outcome of our proposals, even down to the 20th proposal. This should be generally reasonable for a heaviest *n* paths problem, and some prior proposed methods for beam search had found that such heuristics do not regularly impact performance, despite being suboptimal. [12]

Further Modification 3: A variation of Schwartz et al. Figure 5c. We observed that our simulated protein often fluctuates between two states (e.g.: 'ABABAB'), which resembles a large insertion event that can mask real motifs flanking them. We implemented Viterbi with two passes on the full sequence, the first pass to identify pairs of letters that frequently fluctuate between each other ('ABABAB'), and the second to reduce these 'ABABAB' insertions so that they can be compressed into one layer of the trellis, allowing us to see patterns before and after these large insertions. [13]

Evaluation: We compared our methods against a *k*-mer index. Not only is this the most straightforward approach to the same problem, but it will also tell us if some of the paths we proposed in Viterbi really exist in the real sequence, or if they are false constructs of our graph-based approach. We expect the kmer index to perform well when each transition in "ABCDE" (AB, BC, CD) occurs with near 100% probability. However, if just a few of the transitions are < 90%, the probability of finding the full pattern with a *k*-mer index will significantly drop.

Results

To understand the runtime and space associated with each methodology, we benchmarked the performance of each architecture. Notably, our benchmarking software scales efficiently and offers the ability for the user to evaluate our algorithms in any way of interest. However, for the sake of demonstration, we evaluated performance on a 100,000-character sequence and ran each architecture-sequence combination 25 independent times. We explore the performance to make suggestions on the relative performance of each architecture on real-time sequences of protein folding patterns.

In relation to space complexity, we find that the BWT-HMM with merging requires the least space. Interestingly, MLSE Viterbi marginally outperforms the traditional BWT-HMM (the vanilla one without merging) on the space front, with both MLSE Viterbi and traditional BWT-HMM close together and lagging higher in needed space than the BWT-HMM with merging. This advantage of the BWT-HMM with merging on the space front is notable but likely limited in overall usefulness given that the difference in space complexity is on the order of 10 MiB. Given that 1 MiB is about 1.05 MB, we are talking about a difference on the order of 10 MB for a 100,000 character sequence. Such a difference is something but likely not enough of a difference in space to significantly influence the attractiveness of any algorithm on its own. Note that the x-axis on the following graph starts in the vicinity of 300 MiB rather than 0 MiB.

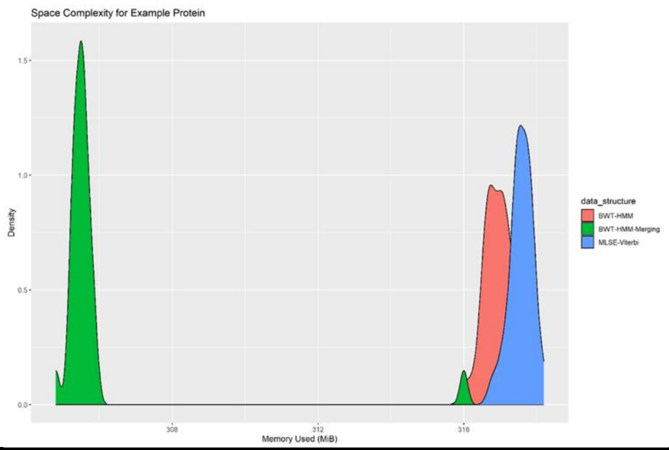


Table 1: Memory Usage by Architecture			
Architecture	Average (MiB)	Median (MiB)	Standard Deviation (MiB)
BWT-HMM	316.924	317.000	0.352
BWT-HMM with Merging	305.896	305.500	2.118
MLSE Viterbi	317.576	317.600	0.310

In relation to time complexity, we find that MLSE Viterbi far outperforms any of the other methods, both in expectation and in variability. The average runtime for MLSE Viterbi was 0.440 seconds which is 0.696 seconds better than that of the BWT-HMM with merging and 6.236 seconds better than the traditional BWT-HMM. These figures represent a 61.3% and 93.4% decrease in expected runtime, respectively. This is a major result at the core of our findings. We find that MLSE Viterbi offers a powerful compression scheme that is in the same ballpark space wise as the BWT-HMM and its variations but significantly faster in terms of runtime.

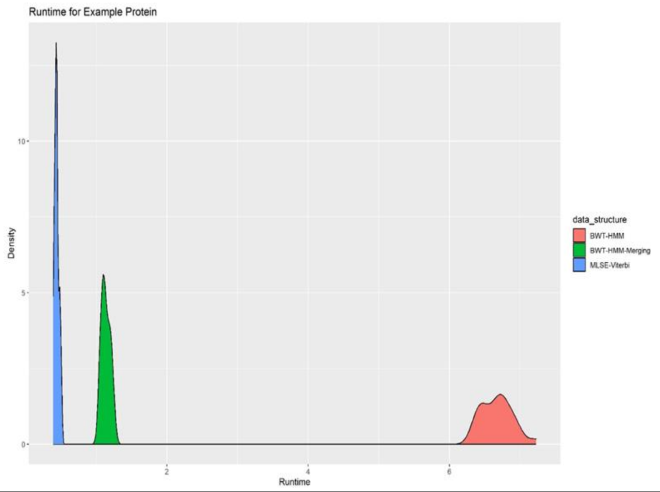


Table 2: Runtime by Architecture			
Architecture	Average (sec)	Median (sec)	Standard Deviation (sec)
BWT-HMM	6.676	6.707	0.218
BWT-HMM with Merging	1.136	1.122	0.062
MLSE Viterbi	0.440	0.438	0.032

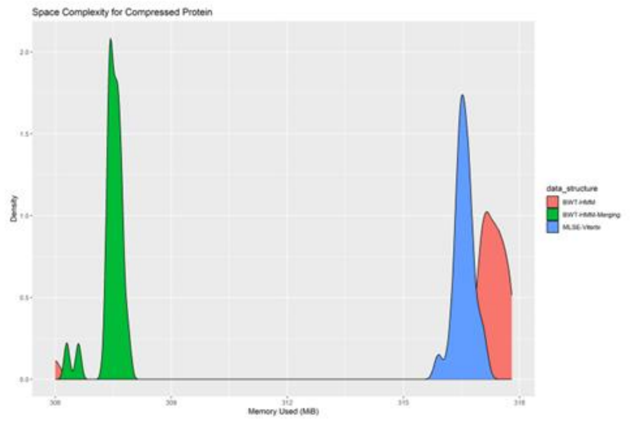
To more robustly quantify our results, we analyzed the performance of MLSE Viterbi in the context of a one-sided t-test to check whether the mean was lower for Viterbi than each of the BWT-HMM and BWT-HMM with merging for both memory usage and runtime. This is reflected in Table 3. To be explicit, the t-test is to test whether the mean of "First Population" is less than "Second Population" and the "Statistical Significance?" column is determined in the context of a 0.05 significance level.

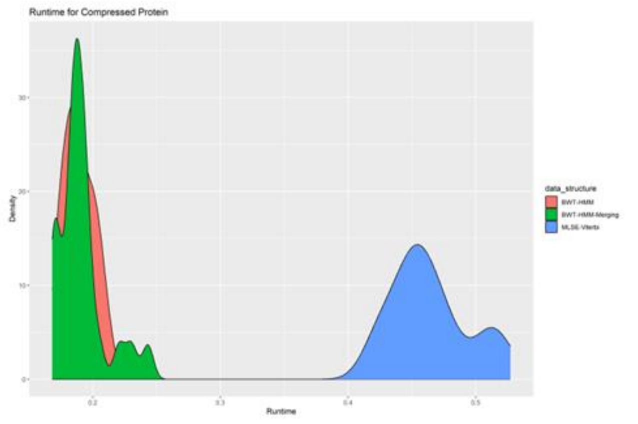
Table 3: Statistical Tests of MLSE Viterbi vs. BWT-HMM-Based Architectures

Metric	First Population	Second Population	Test Statistic	P-Value	Statistical Significance?
Memory Usage	MLSE-Viterbi	BWT-HMM	6.956	1	No
Memory Usage	MLSE-Viterbi	BWT-HMM with Merging	27.28416	1	No
Runtime	MLSE-Viterbi	BWT-HMM	-141.458	3.871×10^{-38}	Yes
Runtime	MLSE-Viterbi	BWT-HMM with Merging	-50.2362	4.272×10^{-35}	Yes

The results are largely by extension no different than before but just contain more statistically robust context. We have statistically significant evidence that there is an advantage to MLSE Viterbi's expected runtime. We also again have evidence that the space usage patterns are relatively comparable across MLSE Viterbi and the other architectures.

As a secondary experiment to analyze the strengths and weakness of MLSE Viterbi in relation to the BWT-HMM-based architectures, we ran benchmarking on a sequence we call a "unique confirmation compression," which in effect is a protein fold cluster sequence compressed such that no two adjacent characters are the same but where run length info that normally would have been tracked for each remaining character has been discarded. This is admittedly a very niche example case, but we thought it would prove to be a useful example of how performance can vary significantly in odd-ball situations at the extremes. We can see that in this extreme and unrealistic case where in effect the passed-in sequence implies a confirmation change at every position we find that the BWT-HMM significantly outperforms MLSE Viterbi in terms of runtime with comparable space complexity outcomes to before. This result is relatively unsurprising given that MLSE Viterbi performs the aforementioned type of compression under the hood, meaning that sequences with repeated cluster states that can be compressed do get advantageously compressed under Viterbi but don't in the case of the BWT-HMM. This specific example represents a case where MLSE Viterbi is losing one of its inherent advantages since we are passing in a sequence that cannot be further compressed using this strategy. We include this to show that the attractiveness of each algorithm unsurprisingly does depend on the mechanics of each sequence, which implies the need to evaluate the attractiveness of each algorithm for unique applications. This is added simply in the spirit of comprehensiveness, with the idea of highlighting both edge cases and things that we considered along the journey.





Big picture, MLSE Viterbi remains significantly more efficient in runtime and comparable in space complexity for the vast majority of applications.

Finally, we tested our various Viterbi modifications on real and simulated sequences. In most cases and across different levels of sequence variation, Viterbi was able to extract the ground truth across its top ~5 proposals, but because we make proposals of a fixed length longer than the real path, the real signals are mixed with noisy sequences. For instance, a real path of 12345 in a proposal length of 10 appears as 1234512345 or X12345XX... in various proposals. We can easily visually infer them in our cases, but in further studies, we would consider applying local alignment or assembly methods to extract the ground truth from our noisy data.

We compared this to the evaluation method of the *k*-mer index, which finds the ground truth when sequence variation is low. The k-mer index is able to find common subsequences that are exact matches, but it struggles to distinguish between slight variations of the same sequence motif, such as ABCD and BCDE. Therefore, Viterbi appears to be a promising method to identify common sequence motifs when some degree of variation is present.

Conclusion

In this project, we sought to enhance the processing and analysis of protein conformational sequences within the broader context of computational protein simulations, crucial for understanding the intricate mechanisms of proteins at an atomic level. Our approach, influenced by the challenges in Molecular Dynamics (MD) simulations, particularly in terms of computational and memory demands, led us to explore the integration of Markov State Models (MSM) with innovative data structures like the Burrows-Wheeler Transform (BWT) and Hidden Markov Models (HMM). This integration aimed to streamline the handling of extensive simulation datasets, making the analysis more accessible and feasible for long-term studies. By refining the traditional MSM approach, which treats protein conformations as discrete states, our method intended to capture a more nuanced and accurate representation of protein dynamics, essential for understanding complex biological processes.

Our work also spotlighted the Viterbi algorithm, renowned for its effectiveness in finding the most probable path in various applications. The algorithm was adapted to identify common sequence motifs within the protein conformational sequences. This adaptation was provided a refined lens through which to view and understand the complexities of protein folding. In terms of time and space complexities, BWT-HMM merging had a marginally smaller space footprint than other methods. However, the strongest benefit offered by any architecture was the stronger runtime of Viterbi.

Testing our models, we demonstrated that innovative takes on foundational genomics data structures and algorithms, like the Burrows Wheeler Transform and Viterbi, can generalize performantly to molecular dynamics simulations. These findings are particularly relevant for large-scale molecular dynamics simulations, where data volume can be a major constraint. These advancements in time and space complexities represent a significant step forward in making protein simulation studies more efficient and feasible, marking a notable achievement in the computational analysis of biological systems. Further research could be explored into how the architectures we proposed perform on even longer sequences and on sequences with a wider range of properties.

In conclusion, our research not only provides proof of concept but also paves the way for future innovations in MD simulations, particularly in real-time applications. The potential enhancement of our models could revolutionize fields like whole-genome sequencing, offering significant improvements in data storage and analysis efficiency while obtaining data in real-time. Our research serves as a foundational step and serves as a starting point for future endeavors in relation to MD simulations.

Responsibilities

Nick was responsible for coding BWT-HMM and BWT-HMM-Merging. In addition, Nick wrote the Introduction, Prior Work, and Methods and Software sections of the report associated with protein simulations, MD, and BWT-HMM, and contributed to the writing of the Abstract and Conclusion. Lily wrote the code for MD that generates simulated sequences, wrote part of Viterbi with Richard, k-mer index for comparison to Viterbi, and contributed to our various Viterbi methods in collaboration with Richard. In addition, Lily wrote the introduction and methods section of the report related to Viterbi as well as references. Richard implemented the Viterbi methods that we collaboratively designed, as well as the code to produce fake sequences. Richard also wrote the methods and prior works section of Viterbi. Tad wrote the code for the BWT-HMM code with Nick. In addition, Tad wrote the Results and Conclusions sections of the report. Tad also created the whole benchmarking software suite and ran all of the benchmarking analyses. He also wrote most of the README.

References

1. Kuhlman, B., & Bradley, P. (2019). Advances in protein structure prediction and design. *Nature Reviews Molecular Cell Biology*, *20*(11), 681–697. https://doi.org/10.1038/s41580-019-0163-x
2. Hollingsworth, S. A., & Dror, R. O. (2018). Molecular Dynamics Simulation for All. *Neuron*, *99*(6), 1129–1143. https://doi.org/10.1016/j.neuron.2018.08.011
3. Husic, B. E., & Pande, V. S. (2018). Markov State Models: From an Art to a Science. *Journal of the American Chemical Society*, *140*(7), 2386–2396. https://doi.org/10.1021/jacs.7b12191
4. Langmead, B., Trapnell, C., Pop, M., & Salzberg, S. L. (2009). Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*, *10*(3), R25. https://doi.org/10.1186/gb-2009-10-3-r25
5. Eddy, S. R. (1998). Profile hidden Markov models. *Bioinformatics*, *14*(9), 755–763. https://doi.org/10.1093/bioinformatics/14.9.755
6. Sulistyawan, I. G. E., Arifin, A., and Fatoni, M. H (2022). An Adaptive BWT-HMM-based Lossless Compression System for Genomic Data. *2020 International Conference on Computer Engineering, Network, and Intelligent Multimedi*a (CENIM), Surabaya, Indonesia, 2020, pp. 429-434, doi: 10.1109/CENIM51130.2020.9297871.
7. Li, Y., & Dong, Z. (2016). Effect of Clustering Algorithm on Establishing Markov State Model for Molecular Dynamics Simulations. *Journal of Chemical Information and Modeling*, *56*(6), 1205–1215. https://doi.org/10.1021/acs.jcim.6b00181
8. Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, *13*(2), 260–269. https://doi.org/10.1109/tit.1967.1054010
9. Forney, G. (1972). Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference. *IEEE Transactions on Information Theory*, *18*(3), 363–378. https://doi.org/10.1109/tit.1972.1054829
10. Kelly Knudson Fitzpatrick, *Magnetic Recording*, Editor(s): Robert A. Meyers, Encyclopedia of Physical Science and Technology (Third Edition), Academic Press, 2003, Pages 945-958, ISBN 9780122274107, doi.org/10.1016/B0-12-227410-5/00394-X.
11. Li, W., Sidorenko, V., Jerkovits, T., & Kramer, G. (2019). On Maximum-Likelihood Decoding of Time-Varying Trellis Codes. *Elib (German Aerospace Center)*. https://doi.org/10.1109/redundancy48165.2019.9003340
12. Xie Lingyun, & Du Limin. (2004). Efficient viterbi beam search algorithm using dynamic pruning. *IEEE Xplore*. https://doi.org/10.1109/icosp.2004.1452759
13. Bouloutas, A., Hart, G. W., & Schwartz, M. (1991). Two extensions of the Viterbi algorithm. *IEEE Transactions on Information Theory*, *37*(2), 430–436. https://doi.org/10.1109/18.75270