molecules

MDPI

Article

Integrating Rigidity Analysis into the Exploration of Protein Conformational Pathways Using RRT* and MC

Fatemeh Afrasiabi *,†, **Ramin Dehghanpoor** † **and Nurit Haspel**

Department of Computer Science, University of Massachusetts Boston, Boston, MA 02125, USA; ramin.dehghanpoor001@umb.edu (R.D.); nurit.haspel@umb.edu (N.H.)

* Correspondence: fatemeh.afrasiabi001@umb.edu

† These authors contributed equally to this work.

Abstract: To understand how proteins function on a cellular level, it is of paramount importance to understand their structures and dynamics, including the conformational changes they undergo to carry out their function. For the aforementioned reasons, the study of large conformational changes in proteins has been an interest to researchers for years. However, since some proteins experience rapid and transient conformational changes, it is hard to experimentally capture the intermediate structures. Additionally, computational brute force methods are computationally intractable, which makes it impossible to find these pathways which require a search in a high-dimensional, complex space. In our previous work, we implemented a hybrid algorithm that combines Monte-Carlo (MC) sampling and RRT*, a version of the Rapidly Exploring Random Trees (RRT) robotics-based method, to make the conformational exploration more accurate and efficient, and produce smooth conformational pathways. In this work, we integrated the rigidity analysis of proteins into our algorithm to guide the search to explore flexible regions. We demonstrate that rigidity analysis dramatically reduces the run time and accelerates convergence.

Keywords: protein conformations; conformational pathways; rapidly exploring random trees algorithm; rigidity analysis

Citation: Afrasiabi, F.; Dehghanpoor, R.; Haspel, N. Integrating Rigidity Analysis into the Exploration of Protein Conformational Pathways Using RRT* and MC. *Molecules* **2021**, *26*, 2329. https://doi.org/10.3390/molecules26082329

Academic Editors: Filip Jagodzinski and Kevin Molloy

Received: 16 March 2021
Accepted: 13 April 2021
Published: 16 April 2021

1. Introduction

Proteins take part in nearly all biological functions in the cell. For this reason, it is of paramount importance to understand how they function. A protein's biological function is dictated by its 3-dimensional structure and its interaction with other molecules. From the structures of proteins, we have derived our understanding both of the functions of individual proteins, and the general principles of protein structure and dynamics. Proteins fold into unique 3D shapes, and as they are dynamic molecules, they may change their shapes as a consequence of environmental changes or binding to other molecules. The rearrangement of a protein's tertiary structure in response to environmental changes, external factors, or binding of a ligand from one conformation to another is called the conformational change. For these reasons, in order to fully elucidate the functionality of proteins, characterizing these conformational changes, which are significant factors in protein properties, has been a continuing hotspot for research [1–4].

However, one of the main challenges is that these conformational changes are transitory and that often makes it hard to characterize the intermediate structures. Researchers have used computational methods to try to capture these computational changes and the paths the proteins take when they transition from one conformation to another. For instance, Molecular Dynamics (MD) simulations [5] can examine protein dynamics up to the microsecond level though they are still not fast enough to capture conformational transitions that take place over larger timescales-milliseconds or more. Methods such as targeted, biased, or steered MD, umbrella sampling, replica exchange, activation relaxation, conformational flooding, swarm methods, and others attempt to lower the computational

demands of MD-based approaches [6–9], and Multiscale Molecular Dynamics Simulation methods [10]. Some methods focus on deforming a trivial conformational path (obtained, for instance, through morphing) to improve its energy profile. MD-based approaches tend to produce one path, depending on the starting conditions. Another class of methods samples 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) [11,12], elastic network [13,14], robotic motion [15] 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 [16]. 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 [17] or multi-scale sampling [18–20]. A brief review of adaptive sampling methods is in [21]. Efficient and accurate sampling of protein conformational pathways helps to better identify intermediate states which also correspond to local energy minima. The SPIRAL method described in [22] 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 [23]. It presents a promising research direction in improving decoy generation for template-free protein structure prediction. A similar approach is proposed in [24] by using a highly parallelized version of RRT called Transition-based RRT. This algorithm also aims at identifying the most probable conformations of a class of highly flexible biomolecules such as Intrinsically Disordered Proteins (IDPs).

In [18], the 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 bias 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 ensemble in the absence of correlations between trajectories. A brief review of the current methods and their comparison can be found in [25].

A different class of methods focus on 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. Methods in this category adapt sampling-based search algorithms [26] developed for the robot motion-planning problem which bears strong analogies to the problem of computing conformations along with 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. Second, the energy surface of a single protein is highly complex 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 [22] 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.

1.1. RRT, RRT, and Adaptive Sampling*

The Rapidly Exploring Random Tree (RRT) algorithm [27] is a robotics-based method that builds a tree at the start configuration by randomly exploring the search space. The algorithm was presented in our previous work, extended by this journal submission [28]. The algorithm is outlined here for clarity.

- Create a random new sample q_r (a new molecular conformation).
- Find q_r's nearest neighbor (q_{near}) on the tree.
- Try to connect q_r and q_{near} by an edge. To do this, we incrementally rotate the neighbor's relevant backbone dihedral angles in the direction of the new conformation.
- Stop when you reach an obstacle (a high energy barrier) or when you reach the newly created sample. The stopping point is q_{new}
- Add q_{new} to the tree with q_{near} as its parent.
- Stop when you reach an RMSD threshold distance from the goal or when the maximum number of iterations was reached.

RRT is known for its ability to rapidly explore the workspace. However, due to its randomness, it expands to all areas of the search space, which means that it does not always converge fast, especially in high-dimensional search spaces like the protein conformational landscape. It also makes the resulting paths from start to goal jagged and "zigzagged".

RRT* [29] is an optimized version of RRT. Due to the random nature of the original RRT's node selection and connection, it does not always produce smooth or optimized paths. RRT* modifies the original algorithm to deliver significantly less ragged and smoother paths to the goal. The optimization is done through a heuristic cost function as follows:

- For every node we also record the tree distance from the root. After finding the nearest neighbor on the tree, we examine the neighborhood of the new node in a fixed radius. If there is a node with a cheaper overall cost, it becomes the node's parent instead of the original nearest neighbor.
- After adding the new node to its cheapest neighbor, the paths are rewired: The neighbors are checked if being rewired to the newly added vertex will decrease their overall cost.
- If the cost indeed decreases, the neighbor is rewired to the newly added vertex. This step makes the path smoother and less jagged looking.

Figure 1 illustrates the process.

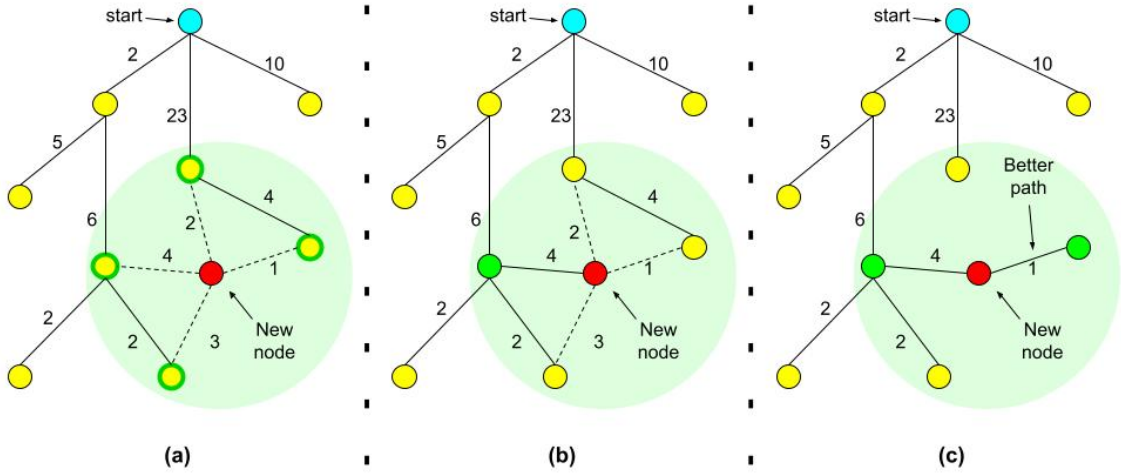

Figure 1. Illustration of RRT* best (least-cost) parent search and the rewiring process. (**a**) A new node is added to the tree and nearest neighbors are highlighted. (**b**) The best parent is chosen among nearest neighbors. (**c**) Costs of rewiring are checked and the tree is rewired.

1.2. Rigidity Analysis

The analysis of rigid residues in proteins can help predict which groups of atoms are more likely to move together in a coordinated way (which are called rigid clusters), and need more energy in order to be destabilized and deformed. These rigid areas of proteins are more likely to stay together during the conformational changes of proteins since structural changes tend to happen around flexible hinges in the protein. Rigidity analysis methods [30,31] use only inter-atomic connectivity and interaction information and do not rely on energy calculations. In our work, we used the Kinari software [30,32] where the rigidity analysis of a protein is done in the following way: During data input and curation (first phase), a PDB file is retrieved from the Protein Data Bank site or uploaded by the user, chains and atoms to be retained are selected, and chemical interactions are calculated and can be removed, added, or modified. During the second phase, modeling options are specified, rigidity analysis is performed, and the results are explored with a script-enhanced Jmol-based (http://www.jmol.org, accessed on 1 March 2021) 3D visualizer.

The Kinari software is written in C++, and its library (KINARI-Lib) implements the pebble game algorithm. It also provides support for bar-joint and body-bar-hinge mechanical models. Pebble game rigidity analysis is an efficient method for extracting rigidity and flexibility information of biomolecules without having to perform costly molecular dynamics simulations [33]. This algorithm works on a multigraph associated with a mechanical model that is constructed from an arbitrary atom-bond network. In the Kinari software, the authors have developed a faster and more robust variation of the pebble game algorithm tailored to the specificities of biopolymers.

1.3. Our Previous Work

In our previous work we implemented a biased Monte Carlo tree search, to explore the conformational space of medium and large proteins [28,34]. In order to decrease the search time and at the same time increase the convergence of the search, we added an optional input argument to allow the algorithm to use a list of residues to manipulate during the search. These residues are assumed to either be co-evolving or otherwise important to the protein structure, function, or dynamics. If no such information is given, any residue may be selected for modification.

At the next step, our goal was to enhance the convergence time even more, as it was still time-consuming for large proteins, and also to generate smooth, even, and least-cost pathways. To do so, we incorporated the RRT* algorithm, the improved and optimized version of the robotics-based Rapidly Exploring Random Tree algorithm, and introduced three key methods, **nearNeighbors**, **chooseParent**, and **rewire** [28]. The **nearNeighbors** method will look for a set of nearest neighbors that lie within an RMSD of at most 1 Å from each anew generated node. Then comes the **chooseParent** method that selects the least-cost node amongst the near neighbors to be the parent of the newly created node. We used an A*-based heuristic cost function to determine the cost of each node (neighbor) and choose the least-cost one. Then the **rewire** method was called to check if we can refine the paths we already had in the tree and make them more straight, as was the goal in RRT*'s creation for robot motion planning [29]. These methods helped the algorithm to generate smoother and less jagged pathways with minimum accumulated costs. At the end, when the tree expands beyond a certain size (dependent on the number of residues), the algorithm prunes the tree by removing the nodes that are within a threshold distance to one another to decrease redundancy.

1.4. This Contribution

This contribution is an extension of our conference paper described in [28]. Instead of using an optional list of residues for manipulation, our goal is to focus explicitly on flexible residues. In this work, we use rigidity analysis to steer and guide the algorithm towards manipulating flexible hinges that are more likely to undergo large-scale changes. Rigidity analysis represents the protein as a set of rigid clusters, connected by flexible

residues. This can assist the algorithm to reduce the randomness of choosing dihedral angles to perturb, and instead, focus on the flexible residues of the protein and select those for perturbing and creating new conformations. Since the proteins undergo large-scale conformational changes, their rigidity properties may change too. Therefore, we run rigidity analysis several times during the search and modify our criteria for selecting residues to manipulate.

2. Materials and Methods

2.1. Description of Algorithm

The input to the algorithm is two PDB coordinates representing conformations of the same protein, denoted as start and end conformations. The goal is to start with the source node, which represents the start conformation, and get as close to the target node, which we denote the end conformation. The algorithm produces pathways made of successive conformations, simulating the path that the protein takes when transitioning between the conformations. The algorithm continues to generate conformations until it reaches a conformation whose RMSD to the end conformation is less than a pre-determined threshold or until a maximum number of iterations is reached. The path is returned as output. An additional input source that we focus on in this work is a list of flexible residues. The list of flexible residues generated by the Kinari software [32] (details mentioned in the next subsection) is used when the algorithm wants to select certain bonds for manipulation. We use semi coarse-grained protein structural representations. Proteins are represented using their backbone chain and C-β atoms (except for Gly, which does not have a C-β atom). The degrees of freedom for each protein molecule are the ϕ and ψ angles around the C-α atoms. This way of representation of the proteins hits a balance between accuracy and speed efficiency.

The algorithm description follows:

During each iteration, a conformation is chosen from the RRT tree to be perturbed. The tree at the beginning only contains the start conformation. We use a biased variable so that one-third of the time the conformation to be perturbed is selected out of the k best conformations. By best, we mean the closest, in terms of RMSD, to the end conformation. In this work, we chose $k = 20$. The rest of the time the conformation is selected from the general pool of nodes, which has all the nodes that are possible to create by manipulating the dihedral angles of molecular structure. The general pool of nodes represents the sampled conformations so far. Once a node is selected for expansion, the algorithm perturbs the molecular conformation of this node at numerous, randomly chosen, ϕ and ψ angles in order to generate a new conformation. These angles are randomly chosen from all the angles of the selected conformation but with the constraint that each particular angle's difference from its counterpart in the end, aka target, conformation is larger than a certain threshold (In this work threshold = 5 degrees). Generally, these angles are selected from all the residues of the conformation, and that's what we did in our previous work. However, in this work, we set aside the rigid residues of the conformation and instead give the algorithm a list of flexible residues to choose these angles from. Since these are the residues that are not part of any rigid clusters, they are more likely to be different in the target conformation and hence are good candidates to be perturbed. After generating the new random conformation by perturbing the angles of flexible residues, it undergoes k steps of steepest descent energy minimization in order to relax local clashes while not altering the structure of the conformation a lot. The constants that we use in this work are $k = 10$ for the steps of steepest descent, and $5 * thenumberofresidues$ as a threshold for acceptable energy. If the new conformation has an energy value above the acceptable threshold after the steps of steepest descent, it is accepted as a new node to our pool of conformations based on a Monte-Carlo criterion. Next, in the nearNeighbors method, the algorithm looks for a set of nearest neighbors (as the RRT* algorithm suggests to find neighbors that exist at a certain distance from the new node) which in our case are the nodes that exist within an RMSD of at most 1 Å from the newly added node. In case there are no other nodes within

this RMSD, the nearest existing node would be the only node selected to create this set of nearest neighbors. Among these nearest neighbors, the best (defined by our cost method) least-cost node is selected to be set as the parent of this new node in the tree. The cost of each node is calculated using an A*-based heuristic cost function. The cost function used here is

$$cost(n) = g(n) + h(n), \tag{1}$$

where $g(n)$ is the summation of the $g(parent(n))$ and the RMSD of n from the start conformation, and $h(n)$ is the RMSD of n from the end conformation. We implemented this cost function to estimate the cost of the cheapest (least-cost) path from the newly generated conformation to the end conformation. The least-cost node is chosen in the chooseParent method and is set as the parent of the new node and added to the main tree. The tree goes through a rewiring procedure afterward, where we check if the newly added node to the tree is a better parent for its neighbors than their current ones. That means, for each node in the set of nearest neighbors, we check if the cost to get that node is less through our newly added node than its current parent. Furthermore, if so, we replace the current parent with the new node.

After multiple iterations, the pool of conformations that we have tends to get larger and larger and it becomes both space and time-wasting to look through all these conformations. Therefore, we go through the pool to find "redundant" nodes, and by that we mean the nodes whose RMSD from each other is 10% or less than their RMSD from the end conformation, and only keep the one closer to the end conformation. This pruning process happens first when the size of the pool tree is equal to the protein molecule's number of residues and then every time it is multiplied by a constant (we chose 3 experimentally). The potential energy function that we use to calculate the energy of a new conformation, in order to decide whether to accept it or not is as follows [35]:

$$E_{total} = E_{HB} + E_{burial} + E_{water} + E_{bond} + E_{angle} + E_{VdW} \tag{2}$$

The first three terms represent hydrogen bonds, burial, and water-mediated interaction terms taken from [35]. The three remaining terms E_{VdW}, E_{bond}, and E_{angle} came from the $AMBER\ ff03$ force field [36]. The E_{VdW} is a Lennard Jones potential that was slightly adapted to tolerate soft collisions between atoms, as a result of the lower resolution. These terms were used to test for clashes and deformities in the structure. A new molecule's potential energy must be smaller than a particular threshold, i.e., k* the number of residues in the molecule, to be accepted to the pool. In this paper, we set $k = 5$. If the energy was too high, we attempted to minimize it using up to 10 steps of gradient-descent minimization on the bond, angle, and VdW terms. If the minimization failed to bring the conformation to within the threshold, it was discarded. If the potential energy was below the threshold, the conformation was accepted to be added to the tree.

The new conformation is now accepted and the next step is to make a new branch (path of conformations) towards it. We do this considering the below criteria:

$$lRMSD_{new} < lRMSD_{parent} \tag{3}$$

$$r < e^{(-\frac{lRMSD_{new} - lRMSD_{parent}}{lRMSD_{new} \cdot a})} \tag{4}$$

where r is a random number in [0,1), and a is a previously-defined constant. a is defined as 0.01 in this work. To avoid getting stuck in the potential local minima, with a small possibility, we allow conformations with lower scores to be accepted.

The iterative sampling of new conformations proceeds until either the score of the generated conformation is below the preset score threshold or the minimum score achieved by the conformations in the pool does not reduce for $M = 500$ iterations. We measure the score of each conformation by its $lRMSD$ with respect to the goal conformation, and the threshold value that we set for this score is within 1.7 and 3.65 based on the protein size and the difficulty of the pathway. This range is a realistic lower bound we found for

our tested proteins since achieving a lower $lRMSD$ is challenging because of sampling errors and variations of the protein molecules The very last step is to extract a path from the tree, from the start conformation to the goal conformation when the threshold $lRMSD$ is reached.

2.2. Integrating Rigidity Analysis

The highlight of this contribution is integrating the rigidity analysis of the protein into our algorithm to steer the algorithm when it selects dihedral angles to manipulate during the search. The information regarding rigidity analysis can be gathered from computational analysis [37,38], experimental information, co-evolution, or rigidity analysis tools such as Kinari [30,32]. For each protein, We used Kinari [30] to obtain the rigid clusters of residues and then extract the flexible residues to be manipulated by our algorithm. The output from Kinari contains the atoms that belong to rigid clusters of any size. In order to avoid very small rigid clusters which may contain noise, we set a threshold for the cluster sizes to be considered. We experimented with thresholds of 6, 7, and 8 atoms. Any residue with atoms not in a rigid cluster was considered flexible. We investigate the effectiveness of using rigidity analysis to guide the residue-selecting procedure in two ways. One approach is to incorporate rigidity analysis only at the beginning of the algorithm as described below:

- We generate the start conformation's rigid clusters of atoms, using a specified threshold for the minimum number of atoms per rigid cluster.
- The flexible residues are then produced using the rigid clusters extracted from the previous step.
- The list of flexible residues of the start conformation is given to the algorithm to be used as candidate residues for perturbation of their ϕ and ψ dihedral angles.

In the above description rigidity analysis is only run once in the beginning, but since the protein undergoes large conformational changes, its rigid clusters may change during the run. The second approach is to use rigidity analysis not only at the beginning but also throughout the algorithm to account for these changes. We hypothesized that feeding the algorithm with the results of rigidity analysis would make the algorithm produce smoother pathways and eventually converge faster. We found that the optimal number of rigidity analysis recalculations was when the conformation changes by more than 2 Å after the previous run. This results in approximately 2–7 runs in the test cases studied here.

2.3. Using the Kinari Software

The Kinari software gets the PDB ID and the chain ID of a protein and returns the set of rigid clusters of atoms. Each rigid cluster is shown as a list of atom numbers. Small clusters with very few atoms (defined by a lower limit) are disregarded because they might represent for example a Benzene ring whose rigidity is of no interest because of the inherent planarity of such structures. We used three different values as the lower limit of atoms in a rigid cluster, 6, 7, and 8, and ran our algorithm for all of these values to see whether this lower limit has a genuine impact on the convergence or the runtime of the algorithm. For instance, by choosing 6 as the lower limit, all of the rigid clusters discovered with fewer than six atoms were not considered.

2.4. Implementation Details

In our previous work, we tested our algorithm on different proteins by submitting jobs on UMass Boston compute cluster, Chimera. Chimera is a heterogeneous distributed memory high-performance compute cluster. However, we were incapable of running the Kinari software on Chimera by submitting jobs through the job scheduler and had to run the algorithm on a new operating system on Chimera without using the job scheduler. Because of this change, we re-ran our previous algorithm [28] on our test cases on the same new OS as the current implementation to be able to compare the results and runtimes properly. Each test case was run ten times and the results were averaged. We used the same RMSD threshold for all runs, and the RMSD results were approximately the same per

protein in all implementations regardless of rigidity analysis. Table 1 shows the comparison of three versions of our algorithm. Furthermore, Table 2 estimates the memory usage by comparing the resulting tree size.

Table 1. Performance and comparison of the results.

Molecule	Path	Init. RMSD †	min RMSD	Time [a]	Time [b]	Time [c]
CaM	$1CLL \rightarrow 1CTR$	14.85	2.94	92.20	118.2	63.02
	$1CTR \rightarrow 1CLL$	14.85	2.93	59.75	67.43	29.19
	$2F3Y \rightarrow 1CFD$	9.93	3	58.67	49.01	37.80
	$1CFD \rightarrow 2F3Y$	9.93	3.57	45	40.38	32
AdK	$1AKE \rightarrow 4AKE$	7.14	2.59	411.67	212.2	182.73
	$4AKE \rightarrow 1AKE$	7.14	2.68	307	186.75	142.69
CVN	$1L5E \rightarrow 2EZM$	15.92	2.93	73.5	68.4	56
	$2EZM \rightarrow 1L5E$	15.92	2.87	33	32.67	29.82
RBP	$2DRI \rightarrow 1URP$	4.08	2.37	26.33	6.67	4.33
	$1URP \rightarrow 2DRI$	4.08	2.42	23	5.75	4.33

† The distances are in Å and the time in minutes. [a] Runtime without rigidity analysis (our previous algorithm). [b] Runtime with rigidity analysis of the start conformation only at the beginning. [c] Runtime with rigidity analysis throughout the algorithm.

Table 2. Comparison of memory usage, size of the trees created.

Molecule	Path	Tree Size †[a]	Tree Size [b]	Tree Size [c]
CaM	$1CLL \rightarrow 1CTR$	2209	2700	2137
	$1CTR \rightarrow 1CLL$	1597	1708	1166
	$2F3Y \rightarrow 1CFD$	1738	1603	1488
	$1CFD \rightarrow 2F3Y$	1432	1410	1354
AdK	$1AKE \rightarrow 4AKE$	2398	1661	1829
	$4AKE \rightarrow 1AKE$	1526	1491	1473
Cvn	$1L5E \rightarrow 2EZM$	3279	2886	3059
	$2EZM \rightarrow 1L5E$	1921	1680	1867
Rbp	$2DRI \rightarrow 1URP$	100	91	84
	$1URP \rightarrow 2DRI$	109	84	63

† Number of nodes in the RRT tree generated. [a] without rigidity analysis (our previous algorithm). [b] with rigidity analysis of the start conformation only at the beginning. [c] with rigidity analysis throughout the algorithm.

3. Results

Below we detail the test case we used and we elaborate on the results for each system separately. Table 1 shows the comparison of three versions of our algorithm: One denoted by (a) for our previous work [28], one denoted by (b) for the version that runs rigidity analysis once in the beginning and one, denoted (c), which runs rigidity analysis throughout the algorithm (see Methods for details). In all cases, the run time improved considerably in implementation (c) with respect to (a). In all but the Calmodulin implementation (b), which performed one rigidity analysis, improved the run time considerably. The memory consumption, measured in the size of the tree, also improved when running rigidity analysis, but the difference is less marked, possibly due to the tree pruning procedures discussed above. Below we detail the results for each protein.

3.1. Test Cases

3.1.1. Calmodulin (CaM)

Calmodulin is a highly conserved, calcium-modulated protein that is found in all eukaryotic cells. It is the most adaptable member of its family, the family of calcium-binding proteins. It acts as a receptor for the Ca^{2+} signal [39]. Calmodulin modulates many cellular effects such as regulation of changes in the concentration of cytosolic calcium ions that control biological processes [40]. It is a relatively small protein consisting of 148 amino acids. Calmodulin has two globular domains that are relatively symmetrical. Each of these domains contains a pair of EF-hand motifs (the EF-hand is a motif that has a helix-loop-helix topology, in which the Ca^{2+} ions are coordinated by ligands within the loop, and it is found in a large family of calcium-binding proteins), and the N- and C-domain motifs that are separated by a flexible linker region. This would make a total of four Ca^{2+} binding sites in this protein.

We considered two Calmodulin pathways for our simulations. First, we simulated the pathway between an open conformation (PDB: $1CLL$) and a closed conformation (PDB:$1CTR$) The RMSD between the two conformations is 14.83 Å. We ran our simulations in both directions, from the open conformation to the closed conformation and then the other way around. Figure 2a,b show the closed ($1CTR$) and open ($1CLL$) conformations, respectively. Furthermore, Figure 3 shows the superimposition of the simulated conformation and the actual conformation of the protein molecule. The flexible residues detected by rigidity analysis are also represented in Figure 2 in blue. As expected, the flexible residues are predicted to be mostly on the linker region between the two EF-hands of the protein. The predicted flexible residues are not exactly the same as the real changed residues when the algorithm gets to the end conformation, but they mostly cover the main flexible parts and guide the algorithm to converge faster towards the end conformation. Table 1 shows a comparison of the results of from our previous work [28], the version that uses the flexible (non-rigid) residues of the start conformation throughout the run, and our current work that uses rigidity analysis to create flexible residues several times throughout the simulation as explained in Section 2.2. For this pathway, when the algorithm calls the Kinari software several times throughout the run, we get the fastest convergence. For instance, for $1CTR \rightarrow 1CLL$ our runtime average is 30.56 min faster than our previous work (Table 1), and for $1CLL \rightarrow 1CTR$, the new runtime is 29.18 min faster. Table 2 displays the size of the RRT tree for our three scenarios. By comparing columns a and c, we can see in all cases, the size of the tree is reduced when we incorporate rigidity analysis a couple of times throughout the algorithm. This denotes a decrease in memory usage. For $1CTR \rightarrow 1CLL$, RRT tree size changed from 1597 to 1166 and for $1CLL \rightarrow 1CTR$, RRT tree size changed from 2209 to 2137 nodes. Figure 4 shows an instance of the conformational path from $1CTR$ to $1CLL$ simulated by the algorithm, with intermediate conformations taken at approximately equal spaces.

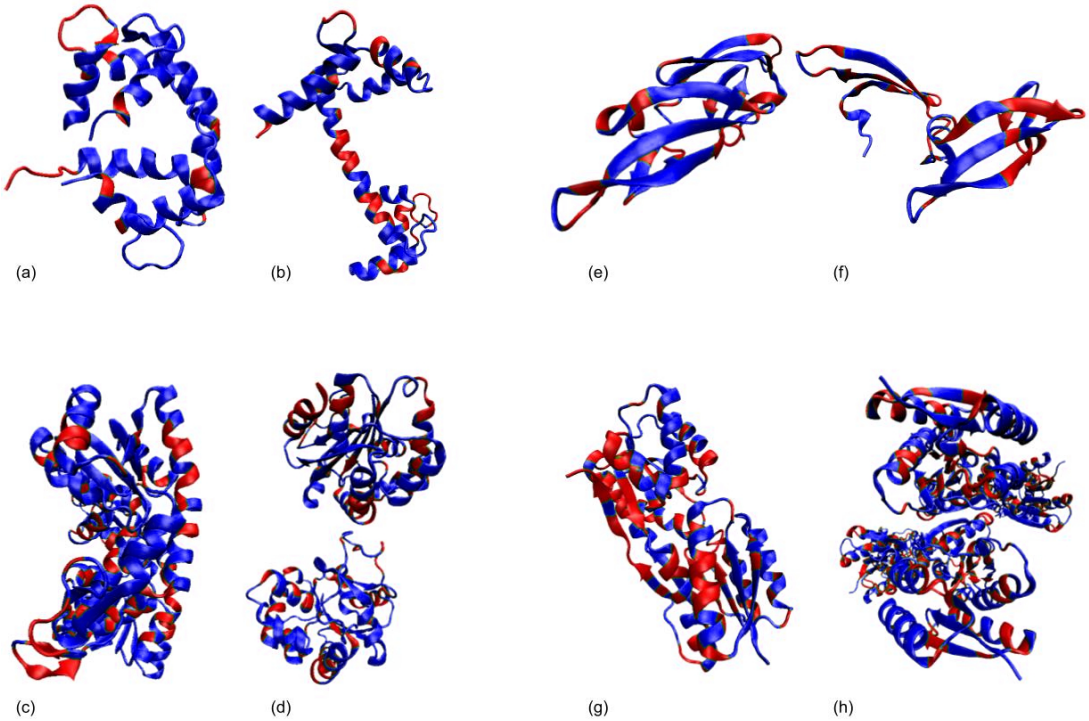

Figure 2. Open and closed conformations of sample proteins. Rigid residues produced by the Kinari software are shown in red, and flexible residues are shown in blue. (**a**) CaM Closed conformation, 1CTR. (**b**) CaM Open conformation, 1CLL. (**c**) AdK Closed conformation, 4AKE. (**d**) AdK Open conformation, 1AKE. (**e**) CVN closed conformation, 2EZM. (**f**) CVN closed conformation, 15LE. (**g**) RBP closed conformation, 2DRI. (**h**) RBP open conformation, 1URP.

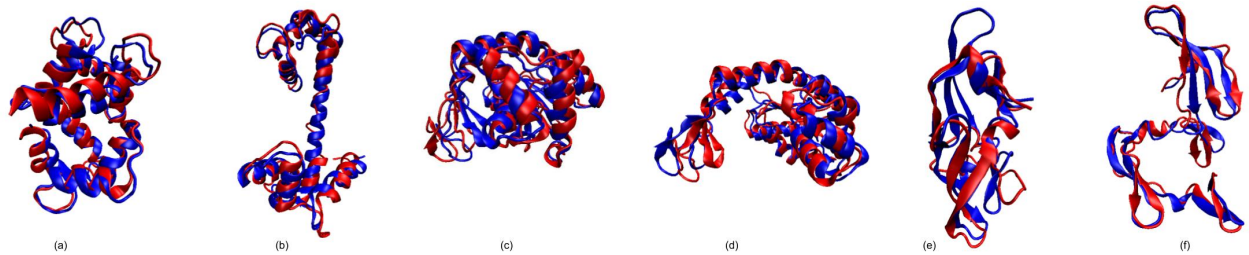

Figure 3. Superimposition of the simulated structure (red) and the original structure (blue). (**a**) 1CTR, (**b**) 1CLL, (**c**) 1AKE, (**d**) 4AKE, (**e**) 2EZM, and (**f**) 1$L5E$.

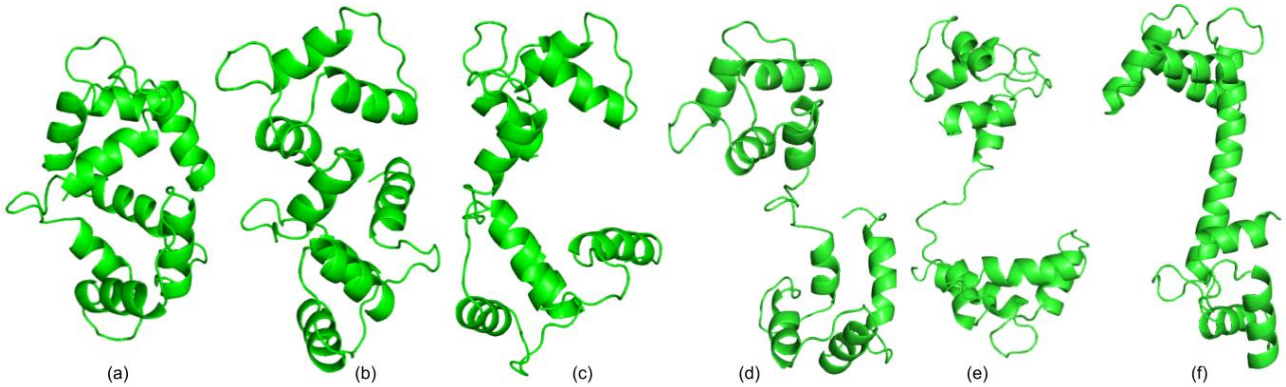

Figure 4. The simulated conformational path for CaM, from 1CTR (**a**) to 1CLL (**f**). Conformations (**b**–**e**) are the intermediate conformations for this path, taken at approximately equal spaces.

The second pathway that we explored for the Calmodulin protein consisted of the Calcium-free form of Calmodulin (PDB: $1CFD$) and Calmodulin/IQ domain complex (PDB: $2F3Y$) conformations. Compared to our previous implementation, the algorithm is on average 13 min faster for $1CFD \rightarrow 2F3Y$, and 20.87 minutes for $2F3Y \rightarrow 1CFD$. The average tree size changed from 1737 to 1488 for $2F3Y \rightarrow 1CFD$, and from 1432 to 1353 for $1CFD \rightarrow 2F3Y$ (Table 2).

3.1.2. Adenylate Kinase (AdK)

AdK is a phosphotransferase enzyme (an enzyme that catalyzes phosphorylation reactions), responsible for catalyzing the phosphoryl transfer between two ADP (adenosine diphosphate) molecules to produce ATP (adenosine triphosphate) and AMP (adenosine monophosphate) [41]. The reason AdK is an important molecule when it comes to conformational changes is that it is a signal-transducing protein, and in order to regulate protein activity, there should exist a balance between conformations [42].

AdK consists of 214 amino acids and has two small domains, LID and NMP, and a CORE domain [43]. The CORE contains residues 1-29, 68-117, and 161-214, the LID domain in residues 118-167, and the NMP domain in residues 30-67. Research suggests that AdK has two relevant conformations; open conformation in the unbound structure, $1AKE$, and a closed conformation $4AKE$, shown in Figure 2d,c, respectively. During the conformational transition from the open conformation to the closed one, the most extensive change happens in LID and NMP domains, while the CORE domain stays relatively rigid. The initial lRMSD between the two conformations is 7.14 Å. We consider the two pathways for the two conformations of AdK; $1AKE \rightarrow 4AKE$, and $4AKE \rightarrow 1AKE$. The average runtime for the $1AKE \rightarrow 4AKE$ path is 228.94 min faster than our previous work (Table 1, refer to Section 2.4 for details on runtime analysis). The average run time for $4AKE \rightarrow 1AKE$ is 164.31 min faster. Figure 5 shows the progression of lRMSD towards the end conformation. The conformational path created by our current algorithm is smoother than our previous one since we attempt to reduce the randomness by focusing on flexible regions. This helps the algorithm to generate more realistic intermediate conformations and converge faster. In Figure 5 the evolution of lRMSD is shown for one example (calculated with and without rigidity analysis) and is not an average. In our future studies, we intend to perform an error analysis to evaluate our method more rigorously. Table 2 depicts how integrating rigidity analysis has downsized the RRT tree from having 1526 nodes to 1473 for the $4AKE \rightarrow 1AKE$ path, and from 2398 to 1829 for the $1AKE \rightarrow 4AKE$. The latter shows a significant amount of memory usage reduction.

3.1.3. Cyanovirin-N (CV-N)

Cyanovirin-N (CV-N) is a potent antiviral fusion inhibitor of HIV and many other viruses [44]. It contains 101 amino acids and has two domains with a 30% sequence identity. The domain-swapped dimer has a higher antiviral affinity than the monomer. With a high energy transition barrier, the two forms can be in a solute dissolved into a solvent. Specific mutations can have an effect on the energy barrier and on stabilizing alternative conformations. We attempt to mimic the unpacking of these repeat domains, from the intertwined monomeric conformation to the domain-swapped dimer conformation, for a single chain. $1L5E$ is the domain-swapped dimer of CV-N in solution, and $2EZM$ is the solution NMR structure of Cyanovirin-N. The closed and open conformations of CVN are displayed in Figure 2e,f, respectively, with the rigid parts of protein shown in red. The two conformations have an initial lRMSD of 15.92 Å. For the $2EZM \rightarrow 15LE$ path, the updated algorithm runs on average 3.18 min faster, and for the $15LE \rightarrow 2EZM$ path, it runs 17.5 min faster on average (Table 1). The evolution of lRMSD towards the end conformation is shown in Figure 5. Like AdK, the new path is smoother.

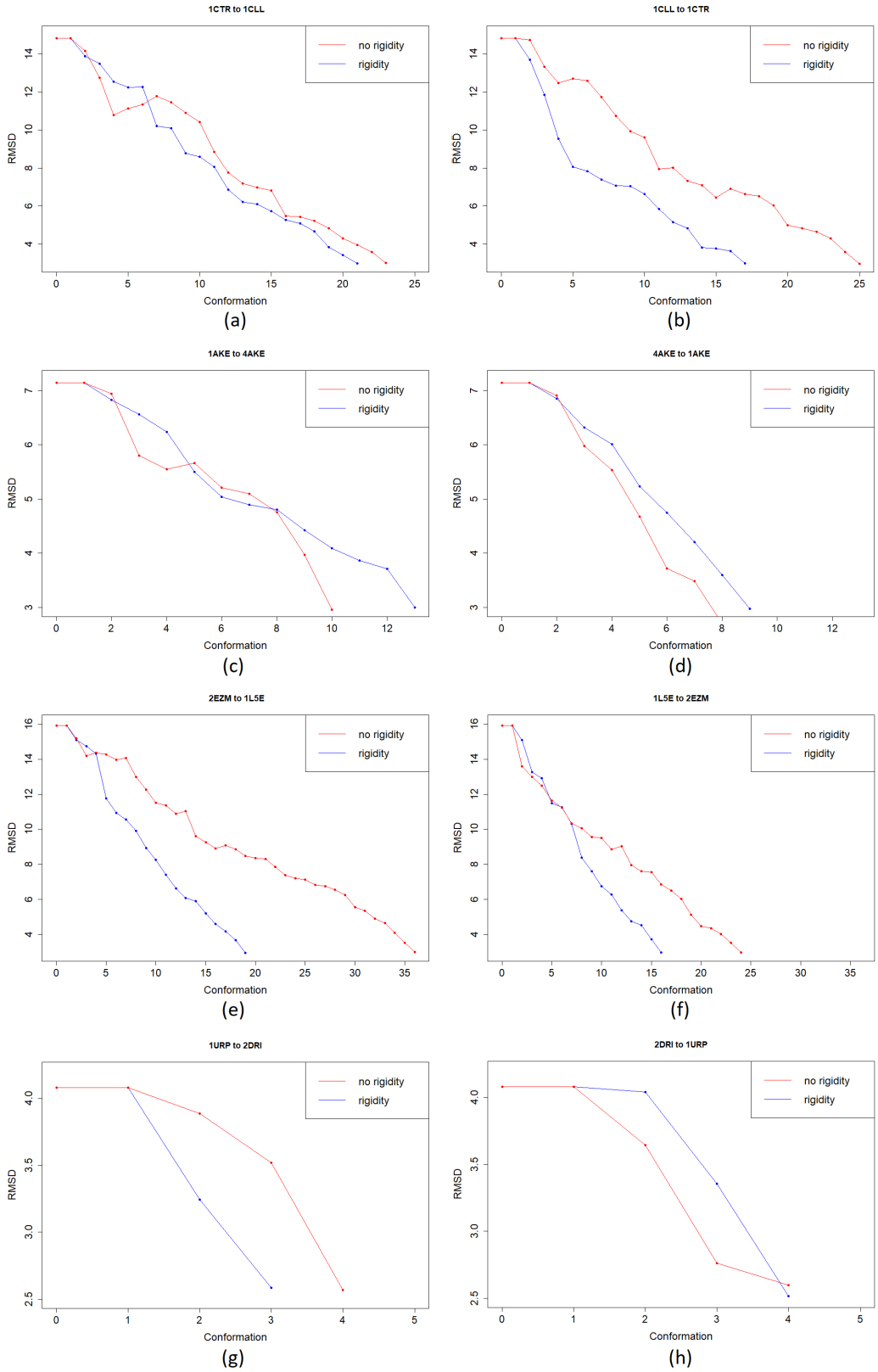

Figure 5. Evolution of lRMSD towards the goal for CaM and AdK. (**a**) $1CTR \rightarrow 1CLL$, (**b**) $1CLL \rightarrow 1CTR$, (**c**) $1AKE \rightarrow 4AKE$, (**d**) $4AKE \rightarrow 1AKE$, (**e**) $2EZM \rightarrow 1L5E$, (**f**) $1L5E \rightarrow 2EZM$, (**g**) $1URP \rightarrow 2DRI$, and (**h**) $2DRI \rightarrow 1URP$. The blue line shows the paths obtained by our updated algorithm using rigidity analysis and the red line shows the paths obtained by our previous work based on RRT* algorithm.

3.1.4. Ribose Binding Protein (RBP)

In order to function correctly, conformational changes are essential in bacterial periplasmic receptors in chemotaxis and transport. These conformational changes allow ligands to enter and exit and enable correct interactions of ligand-bound proteins. We considered two conformations of the Escherichia coli ribose-binding protein, for which the small backbone moves only happen in the hinge region. The secondary structure elements in the two domains and the amino acids in the binding pocket are relatively rigid.

The open (PDB: $1URP$) and closed (PDB: $2DRI$) conformations have dissimilar surfaces in the regions known to be of importance in chemotaxis and transport, that will change their interactions with the membrane components. Conformational changes are necessary for the function of bacterial periplasmic receptors in chemotaxis and transport. Here, we used two conformations of the protein from *Escherichia 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 [45]. 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.

The closed and open conformations of RBP are displayed in Figure 2g,h, respectively, with rigid parts shown in red. For the $2DRI \rightarrow 1URP$ path, the updated algorithm runs on average 21 min faster, and for the $1URP \rightarrow 2DRI$ path, it runs 18.67 min faster on average (Table 1).

4. Discussion

In this project, we present and compare different approaches to using rigidity analysis in order to find proteins' conformational pathways. The use of flexible parts helps us get better results in terms of time, and in most cases, memory usage Table 2. We examine three scenarios. The first one is the same as our previous work (without using rigidity analysis results). For the second one, we start our algorithm with flexible residues as an input to the algorithm to use instead of all the residues. Furthermore, in the last scenario, our algorithm not only uses the flexible residues at the beginning but also a few times during the process. Based on the yielded average of runtimes and tree sizes, the third scenario provides us with the best results concerning runtime and memory. By providing the flexible residues to our algorithm, we help decrease randomness and search mostly among suitable candidates instead of randomly chosen protein residues to manipulate. Consequently, this approach yields smoother paths that are less randomized in a faster time. In our future studies, we aim to compare the intermediate conformations and the conformational pathways that our algorithm generates with available experimental results. We intend to investigate how close these pathways are to actual conformational pathways and how we can improve them.

Author Contributions: Conceptualization, N.H.; methodology, visualization, writing, review, and editing, N.H., F.A., R.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by NSF grant IIS:2031260.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The tests were run on the supercomputing facilities managed by the Research Computing Department at the University of Massachusetts Boston, *Chimera* which is a heterogeneous distributed memory high performance compute cluster (AMD Opteron 6128 processors, 2.0 Ghz).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MDPI	Multidisciplinary Digital Publishing Institute
RRT	Rapidly-Exploring Random Trees
CaM	Calmodulin
AdK	Adenylate Kinase
CVN	Cyanovirin-N
RBP	Ribose-binding Protein
MD	Molecular Dynamics

References

1. Frappier, V.; Chartier, M.; Najmanovich, R.J. ENCoM server: Exploring protein conformational space and the effect of mutations on protein function and stability. *Nucleic Acids Res.* **2015**, *43*, W395–W400. [CrossRef]
2. Duan, Y.; Liu, Y.; Shen, W.; Zhong, W. Fluorescamine Labeling for Assessment of Protein Conformational Change and Binding Affinity in Protein–Nanoparticle Interaction. *Anal. Chem.* **2017**, *89*, 12160–12167. [CrossRef] [PubMed]
3. Mycroft-West, C.; Su, D.; Elli, S.; Li, Y.; Guimond, S.; Miller, G.; Turnbull, J.; Yates, E.; Guerrini, M.; Fernig, D.; et al. The 2019 coronavirus (SARS-CoV-2) surface protein (Spike) S1 Receptor Binding Domain undergoes conformational change upon heparin binding. *bioRxiv* **2020**. [CrossRef]
4. Ghosh, A.; Elber, R.; Scheraga, H.A. An atomically detailed study of the folding pathways of protein A with the stochastic difference equation. *Proc. Natl. Acad. Sci. USA* **2002**, *99*, 10394–10398. [CrossRef]
5. Case, D.; Cheatham, T.; Darden, T.; Gohlke, H.; Luo, R.; Merz, K.M., Jr.; Onufriev, A.; Simmerling, C.; Wang, B.; Woods, R. The Amber biomolecular simulation programs. *J. Comput. Chem.* **2005**, *26*, 1668–1688. [CrossRef] [PubMed]
6. Brokaw, J.B.; Chu, J.W. On the Roles of Substrate Binding and Hinge Unfolding in Conformational Changes of Adenylate Kinase. *Biophys. J.* **2010**, *99*, 3420–3429. [CrossRef]
7. Chen, J.; Wang, J.; Zhu, W. Zinc ion-induced conformational changes in new Delphi metallo-β-lactamase 1 probed by molecular dynamics simulations and umbrella sampling. *Phys. Chem. Chem. Phys.* **2017**, *19*, 3067–3075. [CrossRef]
8. Zhang, Z.; Ehmann, U.; Zacharias, M. Monte Carlo replica-exchange based ensemble docking of protein conformations. *Proteins Struct. Funct. Bioinform.* **2017**, *85*, 924–937. [CrossRef]
9. Karagöz, G.E.; Acosta-Alvear, D.; Nguyen, H.T.; Lee, C.P.; Chu, F.; Walter, P. An unfolded protein-induced conformational switch activates mammalian IRE1. *Elife* **2017**, *6*, e30700. [CrossRef]
10. Guzman, H.V.; Tretyakov, N.; Kobayashi, H.; Fogarty, A.C.; Kreis, K.; Krajniak, J.; Junghans, C.; Kremer, K.; Stuehn, T. ESPResSo++ 2.0: Advanced methods for multiscale molecular simulation. *Comput. Phys. Commun.* **2019**, *238*, 66–76. [CrossRef]
11. Schroeder, G.; Brunger, A.T.; Levitt, M. Combining Efficient Conformational Sampling with a Deformable Elastic Network Model Facilitates Structure Refinement at Low Resolution. *Structure* **2007**, *15*, 1630–1641. [CrossRef]
12. Bauer, J.A.; Pavlović, J.; Bauerová-Hlinková, V. Normal Mode Analysis as a Routine Part of a Structural Investigation. *Molecules* **2019**, *24*, 3293. PMID: 31510014; PMCID: PMC6767145. [CrossRef] [PubMed]
13. Feng, Y.; Yang, L.; Kloczkowski, A.; Jernigan, R. The energy profiles of atomic conformational transition intermediates of adenylate kinase. *Proteins* **2009**, *77*, 551–558. [CrossRef]
14. Hu, G.; Di Paola, L.; Liang, Z.; Giuliani, A. Comparative Study of Elastic Network Model and Protein Contact Network for Protein Complexes: The Hemoglobin Case. *BioMed Res. Int.* **2017**, *2017*, 2483264. [CrossRef]
15. Guieysse, D.; Cortes, J.; Remaud-Simeon, M.; Simeon, T.; Ruiz de Angulo, V.; Tran, V. A path planning approach for computing large-amplitude motions of flexible molecules. *Bioinformatics* **2005**, *21*, 116–125. [CrossRef]
16. Haspel, N.; Luo, D.; González, E. Detecting intermediate protein conformations using algebraic topology. *BMC Bioinform.* **2017**, *18*, 502. [CrossRef]
17. Raveh, B.; Enosh, A.; Furman-Schueler, O.; Halperin, D. Rapid sampling of molecular motions with prior information constraints. *PLoS Comp. Biol.* **2009**, *5*, e1000295. [CrossRef]
18. Al-Bluwi, I.; Vaisset, M.; Siméon, T.; Cortés, J. Modeling protein conformational transitions by a combination of coarse-grained normal mode analysis and robotics-inspired methods. *BMC Struct. Biol.* **2013**, *13*, S2. [CrossRef]
19. Haspel, N.; Moll, M.; Baker, M.; Chiu, W.; Kavraki, L.E. Tracing Conformational Changes in Proteins. *BMC Struct. Biol.* **2010**, *10*, 1–11. [CrossRef]
20. Molloy, K.; Shehu, A. Elucidating the ensemble of functionally-relevant transitions in protein systems with a robotics-inspired method. *BMC Struct. Biol.* **2013**, *13*, S8. [CrossRef]
21. Hruska, E.; Abella, J.R.; Noeske, F.; Kavraki, L.E.; Clementi, C. Quantitative comparison of adaptive sampling methods for protein dynamics. *J. Chem. Phys.* **2018**, *149*, 244119. [CrossRef]
22. Molloy, K.; Shehu, A. A General, Adaptive, Roadmap-Based Algorithm for Protein Motion Computation. *IEEE Trans. Nanobiosci.* **2016**, *15*, 158–165. [CrossRef] [PubMed]
23. Zaman, A.B.; Shehu, A. Balancing multiple objectives in conformation sampling to control decoy diversity in template-free protein structure prediction. *BMC Bioinform.* **2019**, *20*, 211. [CrossRef] [PubMed]

24. Estana, A.; Molloy, K.; Vaisset, M.; Sibille, N.; Simeon, T.; Bernadó, P.; Cortes, J. Hybrid parallelization of a multi-tree path search algorithm: Application to highly-flexible biomolecules. *Parallel Comput.* **2018**, *77*, 84–100. [CrossRef]
25. Chon, L.; Saglam, A.S.; Zuckerman, D.M. Path-sampling strategies for simulating rare events in biomolecular systems. *Elsevier'S Curr. Opin. Struct. Biol.* **2017**, *43*, 88–94. [CrossRef] [PubMed]
26. Ekenna, C.; Thomas, S.; Amato, N.M. Adaptive local learning in sampling based motion planning for protein folding. *BMC Syst. Biol.* **2016**, *10*, 49. [CrossRef]
27. LaValle, S.M.; James, J.; Kuffner, J. Randomized Kinodynamic Planning. *Int. J. Robot. Res.* **2001**, *20*, 378–400. [CrossRef]
28. Afrasiabi, F.; Haspel, N. Efficient Exploration of Protein Conformational Pathways using RRT* and MC. In Proceedings of the ACM-BCB (in CSBW 2020 Workshop), Virtual Event, Atlanta, GA, USA, 21–24 September 2020.
29. Karaman, S.; Frazzoli, E. Sampling-based Algorithms for Optimal Motion Planning. *Int. J. Robot. Res. IJRR* **2011**, *30*, 846–894. [CrossRef]
30. Metlicka, M.; Bygi, M.N.; Streinu, I. Repairing gaps in Kinari-2 for large scale protein and flexibility analysis applications. In Proceedings of the 2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Kansas City, MO, USA, 13–16 November 2017; pp. 577–580. [CrossRef]
31. Luo, D.; Haspel, N. Multi-Resolution Rigidity-Based Sampling of Protein Conformational Paths. In Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics (BCB'13), Wshington, DC, USA, 22–25 September 2013; Association for Computing Machinery: New York, NY, USA; pp. 786–792. [CrossRef]
32. Fox, N.; Jagodzinski, F.; Li, Y.; Streinu, I. KINARI-Web: A server for protein rigidity analysis. *Nucleic Acids Res.* **2011**, *39*, W177–W183. [CrossRef] [PubMed]
33. Nouri Bygi, M.; Streinu, I. Efficient pebble game algorithms engineered for protein rigidity applications. In Proceedings of the 2017 IEEE 7th International Conference on Computational Advances in Bio- and Medical Sciences (ICCABS'17), Orlando, FL, USA, 19–21 October 2017; pp. 1–6.
34. Vajdi, A.; Joshi, A.; Haspel, N. Integrating Co-Evolutionary Information in Monte Carlo Based Method for Proteins Trajectory Simulation. In Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics, Niagara Falls, NY, USA, 7–10 September 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 598–603. [CrossRef]
35. Papoian, G.A.; Ulander, J.; Eastwood, M.P.; Luthey-Schulten, Z.; Wolynes, P.G. Water in protein structure prediction. *Proc. Natl. Acad. Sci. USA* **2004**, *101*, 3352–3357. [CrossRef]
36. Duan, Y.; Wu, C.; Chowdhury, S.; Lee, M.C.; Xiong, G.; Zhang, W.; Yang, R.; Cieplak, P.; Luo, R.; Lee, T.; et al. A point-charge force field for molecular mechanics simulations of proteins based on condensed-phase quantum mechanical calculations. *J. Comput. Chem.* **2003**, *24*, 1999–2012. [CrossRef]
37. Dehghanpoor, R.; Ricks, E.; Hursh, K.; Gunderson, S.; Farhoodi, R.; Haspel, N.; Hutchinson, B.; Jagodzinski, F. Predicting the Effect of Single and Multiple Mutations on Protein Structural Stability. *Molecules* **2018**, *23*, 251. [CrossRef] [PubMed]
38. Shahbazi, Z.; Demirtas, A. Rigidity Analysis of Protein Molecules. *J. Comput. Inf. Sci. Eng.* **2015**, *15*, 031009. [CrossRef]
39. Stevens, F.C. Calmodulin: An introduction. *Can. J. Biochem. Cell Biol.* **1983**, *61*, 906–910. [CrossRef] [PubMed]
40. Hoeflich, K.P.; Ikura, M. Calmodulin in Action: Diversity in Target Recognition and Activation Mechanisms. *Cell* **2002**, *108*, 739–742. [CrossRef]
41. Chang, H.Y.; Fu, C.Y. Adenylate Kinase. In *Encyclopedia of Food Microbiology*, 2nd ed.; Batt, C.A., Tortorello, M.L., Eds.; Academic Press: Oxford, UK, 2014; pp. 18–23. [CrossRef]
42. Schrank, T.; Bolen, D.; Hilser, V. Rational modulation of conformational fluctuations in adenylate kinase reveals a local unfolding mechanism for allostery and functional adaptation in proteins. *Proc. Natl. Acad. Sci. USA* **2009**, *106*, 16984–16989. [CrossRef]
43. Daily, M.D.; Phillips, G.N.; Cui, Q. Many local motions cooperate to produce the adenylate kinase conformational transition. *J. Mol. Biol.* **2010**, *400*, 618–631. [CrossRef]
44. Zappe, H.; Snell, M.; Bossard, M. PEGylation of cyanovirin-N, an entry inhibitor of HIV. *Adv. Drug Deliv. Rev.* **2008**, *60*, 79–87. [CrossRef]
45. Björkman, A.J.; Mowbray, S.L. Multiple open forms of ribose-binding protein trace the path of its conformational change. *J. Mol. Biol.* **1998**, *279*. [CrossRef]