Article

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

Fatemeh Afrasiabi ^{1,†,‡,*} , Ramin Dehghanpoor ^{2,†,‡,*} and Nurit Haspel ^{3,†,‡,*}

¹ fatemeh.afraziabi001@umb.edu

² ramin.dehghanpoor001@umb.edu

³ nurit.haspel@umb.edu

* Correspondence: fatemeh.afraziabi001@umb.edu;

† University of Massachusetts Boston, Boston, Massachusetts

‡ 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

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.

Copyright: © 2021 by the authors. Submitted to *Journal Not Specified* for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

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.

89 1.1. RRT, RRT*, and Adaptive Sampling

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

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

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

108
109 RRT* [29] is an optimized version of RRT. Due to the random nature of the original
110 RRT's node selection and connection, it does not always produce smooth or optimized
111 paths. RRT* modifies the original algorithm to deliver significantly less ragged and
112 smoother paths to the goal. The optimization is done through a heuristic cost function
113 as follows:

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

123 Figure 1 illustrates the process.

124 1.2. Rigidity Analysis

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

138 The Kinari software is written in C++, and its library (KINARI-Lib) implements
139 the pebble game algorithm. It also provides support for bar-joint and body-bar-hinge
140 mechanical models. Pebble game rigidity analysis is an efficient method for extracting

141 rigidity and flexibility information of biomolecules without having to perform costly
142 molecular dynamics simulations [33]. This algorithm works on a multigraph associated
143 with a mechanical model that is constructed from an arbitrary atom-bond network. In
144 the Kinari software, the authors have developed a faster and more robust variation of
145 the pebble game algorithm tailored to the specificities of biopolymers.

146 1.3. Our Previous Work

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

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

170 1.4. This Contribution

171 This contribution is an extension of our conference paper described in [28]. Instead
172 of using an optional list of residues for manipulation, our goal is to focus explicitly on
173 flexible residues. In this work we use *rigidity analysis* to steer and guide the algorithm
174 towards manipulating flexible hinges that are more likely to undergo large-scale changes.
175 Rigidity analysis represents the protein as a set of rigid clusters, connected by flexible
176 residues. This can assist the algorithm to reduce the randomness of choosing dihedral
177 angles to perturb, and instead, focus on the flexible residues of the protein and select
178 those for perturbing and creating new conformations. Since the proteins undergo large-
179 scale conformational changes, their rigidity properties may change too. Therefore, we
180 run rigidity analysis several times during the search and modify our criteria for selecting
181 residues to manipulate.

182 2. Materials and Methods

183 2.1. Description of the algorithm

184 The input to the algorithm is two PDB coordinates representing conformations of
185 the same protein, denoted as start and end conformations. The goal is to start with the
186 source node, which represents the start conformation, and get as close to the target node,
187 which we denote the end conformation. The algorithm produces pathways made of
188 successive conformations, simulating the path that the protein takes when transitioning
189 between the conformations. The algorithm continues to generate conformations until
190 it reaches a conformation whose RMSD to the end conformation is less than a pre-
191 determined threshold or until a maximum number of iterations is reached. The path

192 is returned as output. An additional input source that we focus on in this work is a
193 list of flexible residues. The list of flexible residues generated by the Kinari software
194 [32] (details mentioned in the next subsection) is used when the algorithm wants to
195 select certain bonds for manipulation. We use semi coarse-grained protein structural
196 representations. Proteins are represented using their backbone chain and C- β atoms
197 (except for Gly, which does not have a C- β atom). The degrees of freedom for each protein
198 molecule are the ϕ and ψ angles around the C- α atoms. This way of representation of
199 the proteins hits a balance between accuracy and speed efficiency.

200 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. But 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 $5 * \text{thenumberofresidues}$ 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

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

201 where $g(n)$ is the summation of the $g(\text{parent}(n))$ and the RMSD of n from the start
202 conformation, and $h(n)$ is the RMSD of n from the end conformation. We implemented
203 this cost function to estimate the cost of the cheapest (least-cost) path from the newly
204 generated conformation to the end conformation. The least-cost node is chosen in the
205 chooseParent method and is set as the parent of the new node and added to the main
206 tree. The tree goes through a rewiring procedure afterward, where we check if the newly
207 added node to the tree is a better parent for its neighbors than their current ones. That
208 means, for each node in the set of nearest neighbors, we check if the cost to get that node
209

is less through our newly added node than its current parent. And 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]

$$E_{total} = E_{HB} + E_{burial} + E_{water} + E_{bond} + E_{angle} + E_{VdW} \quad (2)$$

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:

$$lRMSD_{new} < lRMSD_{parent} \quad (3)$$

$$r < e^{\left(-\frac{lRMSD_{new}-lRMSD_{parent}}{lRMSD_{new-a}}\right)} \quad (4)$$

223

224

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 2Å 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. And table 2 estimates the memory usage by comparing the resulting tree size.

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 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

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 motifs¹, 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 14.83Å. We ran our simulations in both directions, from the open conformation to the closed conformation and then the other way around. Figure 2 (a) and (b) show the closed (1CTR) and open (1CLL) conformations, respectively. And 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 works 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 → 1CLL our runtime average is 30.56 minutes faster than our previous work (Table 1), and for 1CLL → 1CTR, the new runtime is 29.18 minutes 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 → 1CLL, RRT tree size changed from 1597 to 1166 and for 1CLL → 1CTR, RRT tree size changed from 2209 to 2137 nodes.

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 minutes faster for 1CFD → 2F3Y, and 20.87 minutes for 2F3Y → 1CFD. The average tree size changed from 1737 to 1488 for 2F3Y → 1CFD, and from 1432 to 1353 for 1CFD → 2F3Y (Table 2).

¹ 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.

343 Adenylate Kinase (AdK)

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

350 AdK consists of 214 amino acids and has two small domains, LID and NMP, and
351 a CORE domain [43]. The CORE contains residues 1-29, 68-117, and 161-214, the LID
352 domain in residues 118-167, and the NMP domain in residues 30-67. Research suggests
353 that AdK has two relevant conformations; Open conformation in the unbound structure,
354 1AKE, and a closed conformation 4AKE, shown in Figure 2, (d) and (c), respectively.
355 During the conformational transition from the open conformation to the closed one, the
356 most extensive change happens in LID and NMP domains, while the CORE domain
357 stays relatively rigid. The initial IRMSD between the two conformations is 7.14Å. We
358 consider the two pathways for the two conformations of AdK; 1AKE → 4AKE, and
359 4AKE → 1AKE. The average runtime for the 1AKE → 4AKE path is 228.94 minutes
360 faster than our previous work (Table 1, Refer to section 2.4 for details on runtime analysis).
361 The average run time for 4AKE → 1AKE is 164.31 minutes faster. Figure 5 shows the
362 progression of IRMSD towards the end conformation. The conformational path created
363 by our current algorithm is smoother than our previous one since we attempt to reduce
364 the randomness by focusing on flexible regions. This helps the algorithm to generate
365 more realistic intermediate conformations and converge faster. Table 2 depicts how
366 integrating rigidity analysis has downsized the RRT tree from having 1526 nodes to 1473
367 for the 4AKE → 1AKE path, and from 2398 to 1829 for the 1AKE → 4AKE. The latter
368 shows a significant amount of memory usage reduction.

369 Cyanovirin-N (CV-N)

370 Cyanovirin-N (CV-N) is a potent antiviral fusion inhibitor of HIV and many other
371 viruses [44]. It contains 101 amino acids and has two domains with a 30% sequence
372 identity. The domain-swapped dimer has a higher antiviral affinity than the monomer.
373 With a high energy transition barrier, the two forms can be in a solute dissolved into a
374 solvent. Specific mutations can have an effect on the energy barrier and on stabilizing
375 alternative conformations. We attempt to mimic the unpacking of these repeat domains,
376 from the intertwined monomeric conformation to the domain-swapped dimer confor-
377 mation, for a single chain. 1L5E is the domain-swapped dimer of CV-N in solution, and
378 2EZM is the solution NMR structure of Cyanovirin-N. The closed and open conforma-
379 tions of CVN are displayed in Figure 2, (e) and (f), respectively, with the rigid parts of
380 protein shown in red. The two conformations have an initial IRMSD of 15.92Å. For the
381 2EZM → 1L5E path, the updated algorithm runs on average 3.18 minutes faster, and for
382 the 1L5E → 2EZM path it runs 17.5 minutes faster on average (Table 1). The evolution
383 of IRMSD towards the end conformation is shown in Figure 5. Like AdK, the new path
384 is smoother.

385 Ribose Binding Protein (RBP)

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

393 The open (PDB: 1URP) and closed (PDB: 2DRI) conformations have dissimilar
394 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 2, (g), and (h), respectively, with rigid parts shown in red. For the *2DRI* → *1URP* path, the updated algorithm runs on average 21 minutes faster, and for the *1URP* → *2DRI* path it runs 18.67 minutes faster on average (Table 1).

3.2. Figures, Tables and Schemes

3.2.1. Tables

Table 1. Performance and Comparison of the Results.

Molecule	Path	Init. RMSD [†]	min RMSD	Time ^a	Time ^b	Time ^c
CaM	1CLL → 1CTR	14.85	2.94	92.20	118.2	63.02
	1CTR → 1CLL	14.85	2.93	59.75	67.43	29.19
	2F3Y → 1CFD	9.93	3	58.67	49.01	37.80
	1CFD → 2F3Y	9.93	3.57	45	40.38	32
AdK	1AKE → 4AKE	7.14	2.59	411.67	212.2	182.73
	4AKE → 1AKE	7.14	2.68	307	186.75	142.69
CVN	1L5E → 2EZM	15.92	2.93	73.5	68.4	56
	2EZM → 1L5E	15.92	2.87	33	32.67	29.82
RBP	2DRI → 1URP	4.08	2.37	26.33	6.67	4.33
	1URP → 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	1C LL $\rightarrow$ 1C TR	2209	2700	2137
	1C TR $\rightarrow$ 1C LL	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

3.2.2. Figures

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.

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) 1L5E.

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

Figure 5. Evolution of IRMSD 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.

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 [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. And 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: All authors contributed equally to the writing. FA and RD conducted the analysis.

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, 2.0Ghz).

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

- 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 Research* **2015**, *43*, W395–W400, [<https://academic.oup.com/nar/article-pdf/43/W1/W395/17435684/gkv343.pdf>]. doi:10.1093/nar/gkv343.
- Duan, Y.; Liu, Y.; Shen, W.; Zhong, W. Fluorescamine Labeling for Assessment of Protein Conformational Change and Binding Affinity in Protein–Nanoparticle Interaction. *Analytical Chemistry* **2017**, *89*, 12160–12167. doi:10.1021/acs.analchem.7b02810.
- Mycroft-West, C.; Su, D.; Elli, S.; Li, Y.; Guimond, S.; Miller, G.; Turnbull, J.; Yates, E.; Guerrini, M.; Fernig, D.; Lima, M.; Skidmore, M. The 2019 coronavirus (SARS-CoV-2) surface protein (Spike) S1 Receptor Binding Domain undergoes conformational change upon heparin binding. *bioRxiv* **2020**. doi:10.1101/2020.02.29.971093.
- Ghosh, A.; Elber, R.; Scheraga, H.A. An atomically detailed study of the folding pathways of protein A with the stochastic difference equation. *Proceedings of the National Academy of Sciences* **2002**, *99*, 10394–10398. doi:10.1073/pnas.142288099.
- Case, D.; Cheatham, T.; Darden, T.; Gohlke, H.; Luo, R.; Jr., K.M.; Onufriev, A.; Simmerling, C.; Wang, B.; Woods, R. The Amber biomolecular simulation programs. *J. Computat. Chem.* **2005**, *26*, 1668–1688.

6. Brokaw, J.B.; Chu, J.W. On the Roles of Substrate Binding and Hinge Unfolding in Conformational Changes of Adenylate Kinase. *Biophysical Journal* **2010**, *99*.
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. *Physical Chemistry Chemical Physics* **2017**, *19*, 3067–3075.
8. Zhang, Z.; Ehmann, U.; Zacharias, M. Monte Carlo replica-exchange based ensemble docking of protein conformations. *Proteins: Structure, Function, and Bioinformatics* **2017**, *85*, 924–937.
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.
10. Guzman, H.V.; Tretjakov, 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. *Computer Physics Communications* **2019**, *238*, 66–76. doi:<https://doi.org/10.1016/j.cpc.2018.12.017>.
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.
12. Bauer, J.A.; Pavlovič, J.; Bauerová-Hlinková, V.
13. Feng, Y.; Yang, L.; Kloczkowski, A.; Jernigan, R. The energy profiles of atomic conformational transition intermediates of adenylate kinase. *Proteins* **2009**, *77*(3), 551–558.
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 Research International* **2017**, *2017*, 2483264. doi:10.1155/2017/2483264.
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. doi:10.1093/bioinformatics/bti1017.
16. Haspel, N.; Luo, D.; González, E. Detecting intermediate protein conformations using algebraic topology. *BMC bioinformatics* **2017**, *18*, 502.
17. Raveh, B.; Enosh, A.; Furman-Schueler, O.; Halperin, D. Rapid sampling of molecular motions with prior information constraints. *Plos Comp. Biol.* **2009**, *5*(2), e1000295.
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.
19. Haspel, N.; Moll, M.; Baker, M.; Chiu, W.; Kavraki, L.E. Tracing Conformational Changes in Proteins. *BMC Struct. Biol.* **2010**, *Suppl1*, S1.
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.
21. Hruska, E.; Abella, J.R.; Noeske, F.; Kavraki, L.E.; Clementi, C. Quantitative comparison of adaptive sampling methods for protein dynamics. *The Journal of Chemical Physics* **2018**, *149*, 244119, [<https://doi.org/10.1063/1.5053582>]. doi:10.1063/1.5053582.
22. Molloy, K.; Shehu, A. A General, Adaptive, Roadmap-Based Algorithm for Protein Motion Computation. *IEEE Transactions on NanoBioscience* **2016**, *15*, 158–165.
23. Zaman, A.B.; Shehu, A. Balancing multiple objectives in conformation sampling to control decoy diversity in template-free protein structure prediction. *BMC Bioinformatics* **2019**, *20*, 211. doi:10.1186/s12859-019-2794-5.
24. Estana, A.; Molloy, K.; Vaisset, M.; Sibille, N.; Simeon, T.; Bernadà, P.; Cortes, J. Hybrid parallelization of a multi-tree path search algorithm: Application to highly-flexible biomolecules. *Parallel Computing* **2018**, *77*, 84 – 100.
25. Chon, L.; Saglam, A.S.; Zuckerman, D.M. Path-sampling strategies for simulating rare events in biomolecular systems. *Elsevier's Current Opinion in Structural Biology* **2017**, *43*, 88–94. doi:<https://doi.org/10.1016/j.sbi.2016.11.019>.
26. Ekenna, C.; Thomas, S.; Amato, N.M. Adaptive local learning in sampling based motion planning for protein folding. *BMC systems biology* **2016**, *10*, 49.
27. LaValle, S.M.; James J. Kuffner, J. Randomized Kinodynamic Planning. *The International Journal of Robotics Research* **2001**, *20*, 378–400, [<https://doi.org/10.1177/02783640122067453>]. doi:10.1177/02783640122067453.
28. Afrasiabi, F.; Haspel, N. Efficient Exploration of Protein Conformational Pathways using RRT* and MC. proc. of ACM-BCB (in CSBW 2020 workshop), 2020.
29. Karaman, S.; Frazzoli, E. Sampling-based Algorithms for Optimal Motion Planning. *International Journal of Robotic Research - IJRR* **2011**, *30*, 846–894. doi:10.1177/0278364911406761.
30. Metlicka, M.; Bygi, M.N.; Streinu, I. Repairing gaps in Kinari-2 for large scale protein and flexibility analysis applications. 2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2017, pp. 577–580. doi:10.1109/BIBM.2017.8217714.
31. Luo, D.; Haspel, N. Multi-Resolution Rigidity-Based Sampling of Protein Conformational Paths **2013**. pp. 787–793.
32. Fox, N.; Jagodzinski, F.; Li, Y.; Streinu, I. KINARI-Web: A server for protein rigidity analysis. *Nucleic acids research* **2011**, *39*, W177–83. doi:10.1093/nar/gkr482.
33. Nouri Bygi, M.; Streinu, I. Efficient pebble game algorithms engineered for protein rigidity applications. 2017 IEEE 7th International Conference on Computational Advances in Bio- and Medical Sciences (ICCABS'17), Orlando, FL **2017**. doi:10.1109/ICCABS.2017.8114301.
34. Vajdi, A.; Joshi, A.; Haspel, N. Integrating Co-Evolutionary Information in Monte Carlo Based Method for Proteins Trajectory Simulation. Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics; Association for Computing Machinery: New York, NY, USA, 2019; pp. 598–603. doi:10.1145/3307339.3343867.

35. Papoian, G.A.; Ulander, J.; Eastwood, M.P.; Luthey-Schulten, Z.; Wolynes, P.G. Water in protein structure prediction. *Proceedings of the National Academy of Sciences* **2004**, *101*, 3352–3357.
36. Duan, Y.; Wu, C.; Chowdhury, S.; Lee, M.C.; Xiong, G.; Zhang, W.; Yang, R.; Cieplak, P.; Luo, R.; Lee, T.; others. A point-charge force field for molecular mechanics simulations of proteins based on condensed-phase quantum mechanical calculations. *Journal of computational chemistry* **2003**, *24*, 1999–2012.
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*. doi:10.3390/molecules23020251.
38. Shahbazi, Z.; Demirtas, A. Rigidity Analysis of Protein Molecules. *Journal of Computing and Information Science in Engineering* **2015**, *15*, 031009. doi:10.1115/1.4029977.
39. Stevens, F.C. Calmodulin: an introduction. *Canadian Journal of Biochemistry and Cell Biology* **1983**, *61*, 906–910. PMID: 6313166, doi:10.1139/o83-115.
40. Hoeflich, K.P.; Ikura, M. Calmodulin in Action: Diversity in Target Recognition and Activation Mechanisms. *Cell* **2002**, *108*, 739–742. doi:https://doi.org/10.1016/S0092-8674(02)00682-7.
41. Chang, H.Y.; Fu, C.Y. Adenylate Kinase. In *Encyclopedia of Food Microbiology (Second Edition)*, Second Edition ed.; Batt, C.A.; Tortorello, M.L., Eds.; Academic Press: Oxford, 2014; pp. 18–23. doi:https://doi.org/10.1016/B978-0-12-384730-0.00003-3.
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. *Proceedings of the National Academy of Sciences of the United States of America* **2009**, *106*, 16984–9. doi:10.1073/pnas.0906510106.
43. Daily, M.D.; Phillips, G.N.; Cui, Q. Many local motions cooperate to produce the adenylate kinase conformational transition. *Journal of molecular biology* **2010**, *400*, 618–631. doi:10.1016/j.jmb.2010.05.015.
44. Zappe, H.; Snell, M.; Bossard, M. PEGylation of cyanovirin-N, an entry inhibitor of HIV. *Advanced drug delivery reviews* **2008**, *60*, 79–87. doi:10.1016/j.addr.2007.05.016.
45. AJ, B.; SL, M. Multiple open forms of ribose-binding protein trace the path of its conformational change. *Journal of Molecular Biology* **1998**, *279*. doi:http://dx.doi.org/10.1006/jmbi.1998.1785.