Adaptive Neighbor Connection Aids Protein Motion Modeling

Chinwe Ekenna, Shawna Thomas, Nancy M. Amato

Abstract—**Robotic motion planning algorithms such as Probabilistic Roadmap Methods (PRMs) have been successful in simulating the protein folding process by building a roadmap, or model, of the folding landscape. This roadmap is constructed by sampling protein conformations and connecting them together with energetically feasible transitions. In this work, we propose an adaptive method to dynamically select an appropriate connection method from a set of connection method candidates. Our framework, Adaptive Neighbor Connection (ANC), learns which strategy to use by examining their success and cost over time. Thus, it frees the user of the burden of selecting the best strategy and allows this selection to change over time. We compare ANC to 6 other distance-based connection methods on a set of 7 well-studied proteins. We show that ANC builds roadmaps quickly with high quality folding pathways.**

I. Introduction

Modeling the protein folding process is crucial to better understanding not only how proteins fold and function, but also how they misfold triggering many devastating diseases. Mad Cow disease and Alzheimer's disease are both associated with protein misfolding and aggregation [5]. Since it is difficult to experimentally observe molecular motions, computational methods for studying this motions become critical.

Traditional computational approaches for generating folding trajectories such as molecular dynamics [15], Monte Carlo methods [7], and simulated annealing [14] provide a single, detailed, high-quality folding pathway at a large computational expense. As such, they cannot be practically used to study global properties of the folding landscape or to produce multiple folding pathways. Statistical mechanical models have been applied to compute statistics related to the folding landscape [19, 4]. While computationally more efficient, these methods do not produce individual pathway trajectories and are limited to studying global averages of the folding landscape.

Robotics-based motion planning techniques, including the Probabilistic Roadmap Method (PRM), have been successfully applied to protein folding [1, 2, 6]. They construct a roadmap, or model, of the motion space by randomly sampling conformations and connecting neighboring ones together with feasible transitions. These robotics-based methods can generate multiple folding pathways in a short amount of time (e.g., a few hours on a desktop PC). This enables the study of both individual folding trajectories and global landscape properties. While promising, making good choices for each of the algorithmic steps remains difficult. Hybrid PRM [10] uses machine learning to dynamically decide how to generate samples. However, the problem of selecting good candidates for conformation connection is still daunting.

In this paper, we apply an adaptive connection framework called Adaptive Neighbor Connection (ANC) to proteins. ANC [8] is a strategy inspired by Hybrid PRM that takes in a list of connection methods and automatically determines the best one to use at a given time. Ideally, ANC should:

- pick a connection method that is most likely to successfully connect samples frequently and punish those that continually do not,
- ensure that all methods have some chance of being picked,
- adapt to changes in performance, and
- consider the time it takes to find and connect neighbors in rewarding/penalizing them.

ANC rapidly learns the best strategy to employ based on a trade-off between success and cost. It monitors each connection method's success and cost and updates a set of selection probabilities accordingly. To apply ANC to proteins, we change the measure of success from number of samples connected to the weight (or quality) of edges created. Thus, ANC will learn which connection methods produce high quality edges efficiently.

We compare the performance of ANC to 6 different distance-based connection methods on a set of 7 well-studied proteins. We examine both time and resulting roadmap quality. Our results confirm that no single connection method is the best choice for all protein inputs. We also show that ANC performs well over the entire set, even though it may not select the best connection method for each individual input.

II. Preliminaries and Related Work

We first describe the protein model used (Section II-A) and then provide an overview of the PRM approach for protein folding (Section II-B). We present various neighbor connection methods in Section II-C and some popular protein distance metrics in Section II-D.

A. Protein Model

A protein is a sequence of amino acids, or residues. We model the protein as a linkage robot where only the ϕ and ψ torsional angles are flexible. This is a standard modeling assumption [17]. Each amino acid has two degrees of freedom, ϕ and ψ. Thus, a protein with n amino acids has $2n$ degrees of freedom.

There are many interactions such as hydrogen bonds and van der Waals interactions [15] that affect the behavior of the protein folding process and can be modeled by a potential energy function. This potential energy function helps quantify how energetically feasible a given protein conformation is. In this work, we employ a coarse-grained potential function from [1]. If the atoms are too close to each other (less than 2.4Å in sampling and 1.0Å in connecting), the energy returned (or given) is high; otherwise, the energy is calculated by:

$$U_{tot} = \sum_{constraints} K_d\{[(d_i - d_0)^2 + d_c^2]^{1/2} - d_c\} + E_{hp}$$

where K_d is 100 kJ/mol, d_i is the length on the ith constraint, and $d_0 = d_c = 2$Å as in [15].

B. PRM for Protein Folding

The Probabilistic Roadmap Method (PRM) [13] is a robotics motion planning algorithm that first randomly samples robot (or protein) conformations, retains valid ones, and connects neighboring samples together with feasible motions (or transitions). To apply PRMs to proteins, the robot is replaced with a protein model and traditional collision detection computations are replaced with potential energy calculations [1, 2, 6].

1) Sampling: Protein conformations, or samples, are randomly generated with bias around the native state, the functional state of the protein and most energetically stable. To do so, samples are iteratively perturbed, starting from the native state, and retained if energetically feasible. A conformation q is added to the roadmap with the following probability:

$$P(\text{accept } q) = \begin{cases} 1 & \text{if } E(q) < E_{min} \\ \dfrac{E_{max} - E(q)}{E_{max} - E_{min}} & \text{if } E_{min} < E(q) \leq E_{max} \\ 0 & \text{if } E(q) > E_{max} \end{cases}$$

where E_{min} is the energy of the open chain and E_{max} is $2E_{min}$. This is the same criteria used in [21].

2) Connection: Once a set of samples is created, they must be connected together with feasible transitions to form a roadmap, or model of the folding landscape. Connecting all possible pairs of samples is computationally infeasible, and it has been shown that only connecting the k-closest neighbors results in a roadmap of comparable quality [18].

Given a pair of samples, we compute a transition between them by a straight-line interpolation of all the ϕ and ψ torsional angles. The transition is retained if all the intermediate conformations along the transition are energetically feasible. We assign an edge weight to reflect the energetic feasibility of the transition as $\sum_{i=0}^{n-1} -log(P_i)$ where P_i is the probability to transit from intermediate conformation c_i to c_{i+1} based on their energy difference $\Delta E_i = E(c_{i+1}) - E(c_i)$:

$$P_i = \begin{cases} e^{\frac{-\Delta E_i}{kT}} & \text{if } \Delta E_i > 0 \\ 1 & \text{if } \Delta E_i \leq 0 \end{cases}$$

where k is the Boltzmann constant and T is the temperature. This allows the most energetically feasible paths to be extracted by standard shortest path algorithms.

C. Candidate Neighbor Selection Methods

Recall that only neighboring (or nearby) samples are attempted for connection because it is infeasible to attempt all possible connections. Typically, conformations that are more similar are more energetically feasible to connect and provide good candidates for connection attempts.

There have been a number of methods proposed for locating candidate neighbors for connection. The most common is the k-closest method which returns the k closest neighbors to a sample based on some distance metric. This can be implemented in a brute force manner taking $O(k \log n)$-time per node, totaling $O(nk \log n)$-time for connection. A similar approach is the r-closest method which returns all neighbors within a radius r of the node as determined by some distance metric. Here, the size of the neighbor set is not fixed but is dependent on the sampling density.

Two randomized variants of these methods are proposed in [18]: KClosest/KRand and RClosest/KRand. For KClosest/KRand, the k_2 closest samples are selected first, and then k neighbors are randomly selected from this set for connection. This introduces some randomness but still bounds the edge length to the distance to the node's k_2-th neighbor. Typically, $k_2 = 3k$. RClosest/KRand selects k random neighbors from those within a distance r. This ensures that k nodes are selected for connection as long as the sampling density is high enough (i.e., there are $\geq k$ nodes within a radius r).

Other methods use data structures to more efficiently compute nearest neighbors. *Metric Trees* [24] organize the nodes in a spatial hierarchical manner by iteratively dividing the set into two equal subsets resulting in a tree with $O(\log n)$ depth. However, as the dataset dimensionality increases, their performance decreases [16]. Thus, their performance will degrade with increasing robot (protein) complexity. *KD-trees* [3] extend the intuitive binary tree into a D-dimensional data structure which provides a good model for problems with high dimensionality. However, a separate data structure needs to be stored and updated each time a node is added to the roadmap.

Approximate neighbor finding methods address the running time issue by instead returning a set of approximate k-closest neighbors. These include spill trees [16], MPNN [25], and Distance-based Projection onto Euclidean Space [20]. These methods usually provide a bound on the approximation error.

D. Distance Metrics

The distance metric plays an important role in determining the best connection to attempt. A distance metric is a function δ that computes some "distance" between two conformations $a = \langle a_1, a_2, \ldots, a_d \rangle$ and $b = \langle b_1, b_2, \ldots, b_d \rangle$, i.e., $\delta(a, b) \rightarrow \mathbb{R}$, where d is the dimension of a conformations. Here, $a_1 \ldots$ and $b_1 \ldots$ are the ϕ and ψ torsional angles for each protein conformations. A good distance metric generally predicts how likely it is that a pair of nodes can be successfully connected. Their success varies and is dependent on the nature of the problem being studied. In this study we look at the following set of distance metrics that are commonly used for motion planning:

Euclidean. The Euclidean distance metric gives equal weighting for all dimensions:

$$\delta_{\text{Eucli}}(a,b) = \sqrt{(a_1-b_1)^2 + (a_2-b_2)^2 + \cdots + (a_d-b_d)^2}.$$

Rigidity Analysis Distance Metric. Rigidity analysis [11, 12] may be used to define a distance metric. A rigidity map, r, marks residue pairs i, j if they have the same rigidity relationship: 2 if they are in the same rigid set, 1 if they are in the same dependently flexible set, and 0 otherwise. Rigidity maps provide a convenient way to define a rigidity distance metric, $r_{\text{dist}}(a,b)$, between two conformations a and b where n is the number of residues:

$$\delta_{\text{Rig}}(a,b) = \sum_{0 \le i < j \le 2n} (r_a(i,j) \neq r_b(i,j)).$$

More details about this distance metric can be found in [23].

Cluster Rigidity Distance Metric. The cluster distance metric is very similar to the rigidity analysis distance metric but it instead only marks residue pairs if they belong to the same rigid cluster. See [23] for more details.

Root Mean Square Distance. The protein model has 6 atoms for each amino acid, namely $C, C(\alpha), R, O, H$. Thus a protein with n amino acids will have $6n$ atoms. Denoting the coordinates of these atoms as x_1 to x_{6n}, the root mean square distance (RMSD) between conformations a and b is

$$\delta_{\text{RMSD}}(a,b) =$$

$$\sqrt{\frac{(x_1^a - x_1^b)^2 + (x_2^a - x_2^b)^2 + ... + (x_{6n}^a - x_{6n}^b)^2}{6n}}.$$

Least RMSD (lRMSD) is the minimum RMSD over all rigid body superpostions of a and b. We use lRMSD in this work.

Knot Theory Distance Metric. This metric examines the topological similarity and differences between conformations [9] by looking at crossing numbers. The resulting distance is

$$\delta_{\text{KT}}(a,b) = 1/4\pi \sum_i \sum_j A_{ij}$$

where A_{ij} is the area on the sphere covered by vectors created between a and b. See [9] for more details.

E. Adaptive Neighbor Connection

Adaptive Neighbor Connection (ANC) generates a set of neighbors to a candidate sample q for connection using a list of connection methods $cm_1, cm_2, \ldots, cm_m$. ANC learns a selection probability p_i for each CM based on its prior success and cost for the given input problem. As each CM is used, ANC updates the probabilities accordingly. This approach is similar to the adaptive approach for selecting sampling methods in Hybrid PRM [10], Algorithm 1 gives a description of ANC during roadmap construction. ANC naturally favors CMs with good performance and invokes them more frequently.

Algorithm 1 ANC

```
Input. A connecting vertex $q$, a set of connection methods $CM$, a local planner $lp$ and a graph $G$.
Output. A connected graph $G$ where additional edges are added over the input graph.
Require: Let $P$ be a set of probabilities such that $p_i$ is the probability selecting $cm_i$ as the connection method. Initialize $p_i = 1/|CM|$, $\forall p_i \in P$.
 1: Randomly pick a connection method $cm_i$ according to $P$
 2: $N = cm_i$.FIND_NEIGHBORS$(q, G)$
 3: for each $n \neq q \in N$ do
 4:   if $lp$.IS_CONNECTABLE$(q, n)$ then
 5:     $G$.ADD_EDGE$(q, n)$
 6:   end if
 7: end for
 8: Let $r$ be the success rate of $lp$ over $N$
 9: Let $c$ be the cost incurred
10: Update $(P, r, c)$ according to Equation 5 and Equation 4
```

III. Learning Selection Probabilities

ANC learns the best CM to use based on the performance of each CM over time. If successful connections increase as a result of the neighbors provided to the local planner, the CM gets rewarded and its probability of getting chosen during the next connection attempt increases. Otherwise, it gets punished and its probability decreases. We also adjust its probability based on the cost. This cost inversely affects the selection probability. Potential energy computations take up a large portion of the total computation time and thus is a good measure of cost. In the results presented here, we calculate the cost as the number of potential energy calls incurred by the local planner.

ANC maintains a weight for each CM similar to Hybrid PRM [10]. These weights keep track of the past performance of each CM. ANC initializes each weight w_i to 1. Based on the weights, ANC computes in a step wise manner a probability p_i^* for cm_i that is insensitive to the change in the cost:

$$p_i^* = (1-\gamma)\frac{w_i(t)}{\sum_{j=1}^{m} w_j(t)} + \gamma\frac{1}{m}, i = 1, 2, ..., m, \qquad (1)$$

where $w_i(t)$ is the weight of cm_i in step t, t is the number of connection attempts made, and γ is a fixed constant. The probability p_i^* is a weighted sum of the relative weight of cm_i and the uniform distribution. This ensures that each CM has some chance of being selected.

Let x_i be the reward for the cm_i that was selected.

$$x_i = \alpha + (1-\alpha)(1 - \frac{y_i(t) - miny_i(t)}{maxy_i(t) - miny_i(t)}) \qquad (2)$$

where $y_i(t)$ = current edge weight, $miny_i(t)$ = minimum edge weight recorded during the current timestep. $maxy_i(t)$ = maximum edge weight recorded during the current timestep and α = a constant value used to normalize the reward. All other rewards for that time step are 0. The reward is thus a function of the edge quality (weight) and the local planner success and attempts. To update the weights, we first take into account an adjusted reward that is not dependent on the cost accrued (calculated as the cost insensitive probability):

$$x_i^* = x_i / p_i^*, i = 1, 2, ...m. \quad (3)$$

Then we update the weights for all the connection methods:

$$w_i(t+1) = w_i(t) \exp \frac{\gamma x_i^*}{m}, i = 1, 2, ...m. \quad (4)$$

The new weight is the current weight multiplied by a factor that depends on the reward received. The exponential factors enable the weights adapt quickly.

We now include the cost in the selection probability:

$$p_i = \frac{\frac{p_i^*}{c}_i}{\sum_{j=1}^{m} w_j(t) \frac{p_i^*}{c}_j}, i = 1, 2, ...K, \quad (5)$$

where c_i is the average cost of attempting to connect i. Thus, a high cost CM has a smaller selection probability.

IV. Experimental Results

We study how ANC performs in practice on several small proteins in comparison to selecting a single CM for the entire execution. For each protein studied, we incrementally construct a roadmap as outlined in Sections II-B until the distribution of folding pathway types (as defined by their secondary structure formation order) has stabilized. See [22] for details on the incremental construction and pathway type stabilization process.

We study several different proteins of varying size and structure, see Table I. We compare the performance of 6 distance-based CMs using the distance metrics defined in Section II-D against the performance of ANC with these 6 CMs as input.

We measure both the time to construct a stable roadmap and the quality of the folding pathways it contains. We quantify the quality of folding pathways as the weight of each edge (i.e., energetic feasibility) times the dominance of that edge (i.e., the number of folding pathways that traverse that edge). Recall that ANC uses both edge quality and cost to learn CM performance. Figure 1 summarizes the results. (For both metrics, lower values are better.)

In all cases, each method produced the same secondary structure formation order in its dominant folding pathways as seen in experiment (given in Table I). Thus, the overall approach is robust to this ordering.

From Figure 1, it is clear that there is not one single best method for every input. For example, KClosest-Euclidean is the fastest for 1PGA but slower than average for NUG2 while KClosest-Rigidity is the fastest for NUG2 but slower than average for 2PTL. Interestingly, these three proteins have the same secondary structure makeup and similar overall folds. Thus, performance cannot be predicted based solely on these properties of the native state. A similar observation may be made about the resulting roadmap quality. For instance, KClosest-KnotTheory produces the best quality for 1PGA but one of the worst quality roadmaps for NUG2. Not only does performance vary widely, it is also difficult to predict.

While specific performance from protein to protein is in general unstable, there are some general trends that appear. As expected, there is a tradeoff between time and quality. The slower methods tend to produce the highest quality roadmaps. In fact, KClosest-KnotTheory and KClosest-lRMSD are consistently the two slowest methods for every protein but produce higher quality roadmaps on average.

ANC is able to adapt to the different protein inputs and make selections that while may not be the best choice in terms of time and quality for every protein, it is almost never the worst where the worst performance can be quite poor. (The only instance where it is the worst performance is in terms of quality for 1PGA. However, for this protein and metric, its performance is similar to 4 of the 6 individual methods.) We see this adaptation clearly when looking at ANC's final selection probabilities for each individual method, see Table II. For example, ANC learns to use KClosest/KRand-Euclidean for 1PGA, and this method has good individual performance in terms of time and moderate performance in terms of quality (see Figure 1). However, for NUG2 ANC learns not to use this method as its individual performance is much poorer (see Figure 1. It never selects KClosest-KnotTheory or KClosest-lRMSD with large probabilities due to their high cost.

Finally, we look at the cumulative performance of each method across all protein studied. Figure 2 shows the percentage difference between each method's performance across all proteins and the best total performance across all proteins for both time and quality. Overall, KClosest-Euclidean is the fastest and KClosest-lRMSD produces the highest quality paths, thus the percentage difference for these two methods on these two metrics is 0. However, neither method is the best performing on both metrics. KClosest-Rigidity and ANC balance time and quality the best, and in many cases significantly better than other methods (e.g., KClosest-Cluster and KClosest/KRand-Euclidean).

V. Conclusion

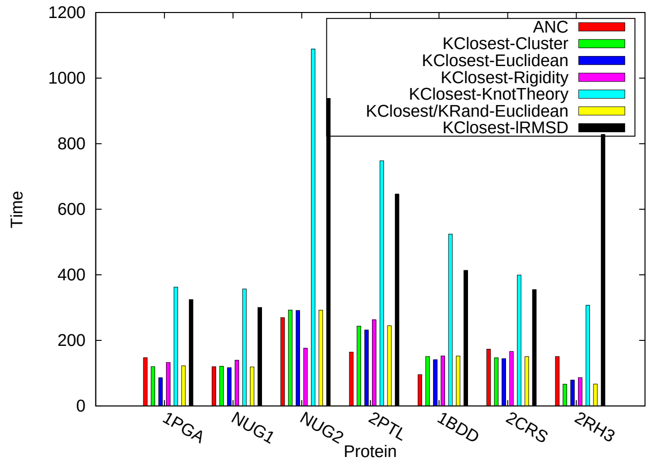
In this work, we present an adaptive method to select appropriate connection methods in the context of PRM roadmap construction. ANC monitors the performance and cost of various connection methods and adjusts their selection probabilities accordingly. The result is an algorithm that can select appropriate methods for different inputs. We compared ANC performance in terms of time and resulting roadmap quality to 6 different distance-based connection methods. ANC, while not always the best performing method for any individual input, performs well over the entire set.

Acknowledgments

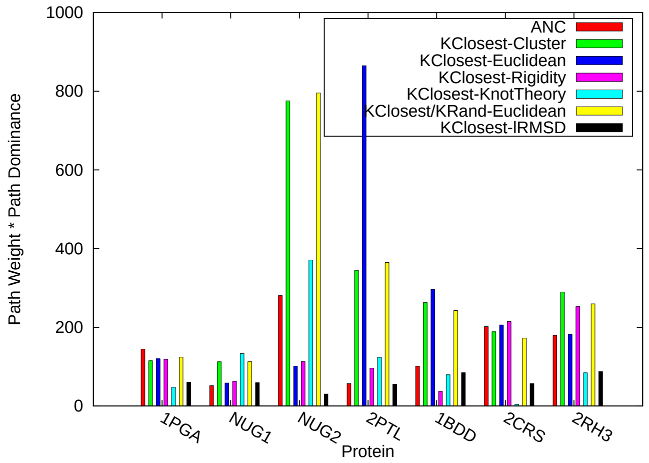
This research supported in part by NSF Grants EIA-0103742, ACR-0081510, ACR-0113971, CCR-0113974, ACI-

Protein	PDB ID	# Residues	Secondary Structure Makeup	Experimental Formation Order
G	1PGA	56	$1\alpha + 4\beta$	[α,β1,β3,β4],β2[1] [α,β4],[β1,β2,β3][2]
G Variant 1	NuG1	56	$1\alpha + 4\beta$	β1-2,β3-4[3]
G Variant 2	NuG2	56	$1\alpha + 4\beta$	β1-2, β3-4[3]
A	1BDD	60	3α	[α2,α3],α1[1] [α1,α2,α3][2]
L	2PTL	62	$1\alpha + 4\beta$	[α,β1,β2,β4],β3[1] [α,β1],[β2,β3,β4][2]
Cardiotoxin analogue III (CTXIII)	2CRS	60	6β	Unknown
Agrobacterium tumefaciens VirC2	2RH3	121	$4\alpha + 2\beta$	Unknown

TABLE I
PROTEINS STUDIED AND THEIR SECONDARY STRUCTURE FORMATION ORDER FROM: [1]HYDROGEN OUT-EXCHANGE EXPERIMENTS [2]PULSED LABELING/COMPETITION EXPERIMENTS AND [3]Φ-VALUE ANALYSIS BRACKETS INDICATE NO CLEAR ORDER.



(a) Time



(b) Quality

Fig. 1. Performance of each individual CM and ANC in terms of (a) time and (b) roadmap quality.

TABLE II
ANC'S FINAL SELECTION PROBABILITIES FOR EACH PROTEIN. BOLDFACE ENTRIES INDICATE THE WINNING PROBABILITY IN EACH.

PDB ID	KClosest-Cluster	KClosest-Euclidean	KClosest-Rigidity	KClosest-KnotTheory	KClosest/KRand-Euclidean	KClosest-lRMSD
1PGA	0.238	0.313	0.175	0.025	**0.089**	0.160
NUG1	**0.833**	0.060	0.027	0.010	0.040	0.030
NUG2	0.009	0.023	**0.937**	0.004	0.013	0.013
2PTL	0.022	**0.903**	0.034	0.004	0.018	0.018
1BDD	0.017	0.011	0.017	0.006	**0.935**	0.015
2CRS	0.033	0.137	**0.757**	0.014	0.027	0.031
2RH3	0.037	**0.836**	0.049	0.009	0.043	0.024

0326350, CRI-0551685, CCF-0833199, CCF-0830753, by the DOE, Chevron, IBM, Intel, HP, by King Abdullah University of Science and Technology (KAUST) Award KUS-C1-016-04, and by the Sclumberger Faculty for the Future Fellowship.

REFERENCES

[1] N. M. Amato and G. Song. Using motion planning to study protein folding pathways. *J. Comput. Biol.*, 9(2):149–168, 2002. Special issue of Int. Conf. Comput. Molecular Biology (RECOMB) 2001.

[2] M.S. Apaydin, D.L. Brutlag, C. Guestrin, D. Hsu, and J.-C. Latombe. Stochastic roadmap simulation: An efficient representation and algorithm for analyzing molecular motion. In *Proc. Int. Conf. Comput. Molecular Biology (RECOMB)*, pages 12–21, 2002.

[3] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998.

[4] Joseph D. Bryngelson and Peter G. Wolynes. Spin glasses and the statistical mechanics of protein folding. *Proc. Natl. Acad. Sci. USA*, 84:7524–7528, 1987.

[5] F. Chiti and C. M. Dobson. Protein misfolding, functional amyloid, and human disease. *Annu. Rev. Biochem.*, 75:333–366, 2006.

[6] J. Cortés, T. Siméon, M. Remaud-Siméon, and V. Tran. Geometric algorithms for the conformational analysis of long protein loops. *J. Computat. Chem.*, 25 (7):956–967, 2004.

[7] David G. Covell. Folding protein α-carbon chains into compact forms by Monte Carlo methods. *Proteins: Struct. Funct. Bioinf.*, 14(3):409–420, 1992.

[8] Chinwe Ekenna, Shawna Thomas, Sam Ade Jacobs, and Nancy Amato. Adaptive neighbor connection for prms, a natural fit for heterogeneous environments and parallelism. Technical Report TR13-006, Texas A&M, May 2013.

[9] Michael A. Erdmann. Protein similarity from knot theory and geometric convolution. In *Proceedings of the eighth annual international conference on Resaerch in computational molecular biology*, RECOMB '04, pages 195–204, New York, NY, USA, 2004. ACM. ISBN 1-58113-755-9. doi: http://doi.acm.org/10.1145/974614.974641. URL http://doi.acm.org/10.1145/974614.974641.

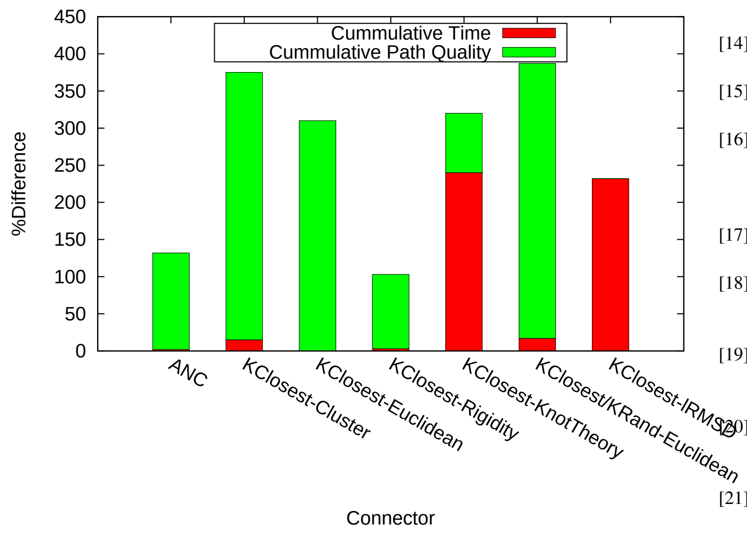


Fig. 2. The percentage difference between each method and the best total performance across all proteins for both time and quality.

[10] D. Hsu, G. Sánchez-Ante, and Z. Sun. Hybrid PRM sampling with a cost-sensitive adaptive strategy. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3885–3891, 2005.

[11] D.J. Jacobs. Generic rigidity in three-dimensional bond-bending networks. *J. Phys. A: Math. Gen.*, 31:6653–6668, 1998.

[12] D.J. Jacobs and M.F. Thorpe. Generic rigidity percolation: The pebble game. *Phys. Rev. Lett.*, 75(22):4051–4054, 1995.

[13] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Automat.*, 12(4):566–580, August 1996.

[14] M. Levitt. A simplified representation of protein conformations for rapid simulation of protein folding. *J. Mol. Biol.*, 104:59–107, 1976.

[15] M. Levitt. Protein folding by restrained energy minimization and molecular dynamics. *J. Mol. Biol.*, 170:723–764, 1983.

[16] T. Liu, A. W. Moore, A. Gray, and K. Yang. An investigation of practical approximate nearest neighbor algorithms. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, pages 825–832, Cambridge, Massachusetts, 2005. MIT Press.

[17] S. Matysiak and C. Clementi. Minimalist protein model as a diagnostic tool for misfolding and aggregation. *J. Mol. Biol.*, 363(1):297–308, 2006.

[18] Troy McMahon, Sam Jacobs, Bryan Boyd, Lydia Tapia, and Nancy Amato. Evaluation of the k-closest neighbor selection strategy for prm construction. Technical Report TR12-002, Texas A&M, College Station Tx., 2011.

[19] Victor Muñoz, Eric R. Henry, James Hoferichter, and William A. Eaton. A statistical mechanical model for β-hairpin kinetics. *Proc. Natl. Acad. Sci. USA*, 95(11):5872–5879, 1998.

[20] E. Plaku and L.E. Kavraki. Quantitative analysis of nearest-neighbors search in high-dimensional sampling-based motion planning. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, July 2006.

[21] G. Song and N. M. Amato. Using motion planning to study protein folding pathways. In *Proc. Int. Conf. Comput. Molecular Biology (RECOMB)*, pages 287–296, 2001.

[22] Shawna Thomas, Xinyu Tang, Lydia Tapia, and Nancy M. Amato. Simulating protein motions with rigidity analysis. In *Proceedings of the 10th annual international conference on Research in Computational Molecular Biology*, RECOMB'06, pages 394–409, Berlin, Heidelberg, 2006. Springer-Verlag.

[23] Shawna Thomas, Xinyu Tang, Lydia Tapia, and Nancy M. Amato. Simulating protein motions with rigidity analysis. *J. Comput. Biol.*, 14(6):839–855, 2007. Special issue of Int. Conf. Comput. Molecular Biology (RECOMB) 2006.

[24] J. K. Uhlmann. Satisfying general proximity/similarity queries with metric trees, 1991.

[25] A. Yershova and S. M. LaValle. Improving motion-planning algorithms by efficient nearest-neighbor searching. *IEEE Trans. Robot. Automat.*, 23(1):151–157, 2007.