A Reinforcement-Learning-Based Approach to Enhance Exhaustive Protein Loop Sampling

Amélie Barozet, Kevin Molloy, Marc Vaisset, Thierry Simeon, Juan Cortés

A Reinforcement-Learning-Based Approach to Enhance Exhaustive Protein Loop Sampling

Amélie Barozet [1,2,*], Kevin Molloy [3], Marc Vaisset [1], Thierry Siméon [1] and Juan Cortés [1,*]

[1]LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
[2]Sanofi recherche & développement, Integrated Drug Discovery, Molecular Design Sciences, 13 quai Jules Guesde, BP 14, 94403 Vitry-sur-Seine Cedex, France
[3]James Madison University, Harrisonburg, Virginia, USA

*To whom correspondence should be addressed.

Abstract

Motivation: Loop portions in proteins are involved in many molecular interaction processes. They often exhibit a high degree of flexibility, which can be essential for their function. However, molecular modeling approaches usually represent loops using a single conformation. Although this conformation may correspond to a (meta-)stable state, it does not always provide a realistic representation.
Results: In this paper, we propose a method to exhaustively sample the conformational space of protein loops. It exploits structural information encoded in a large library of three-residue fragments, and enforces loop-closure using a closed-form inverse kinematics solver. A novel reinforcement-learning-based approach is applied to accelerate sampling while preserving diversity. The performance of our method is showcased on benchmark datasets involving 9-, 12- and 15-residue loops. In addition, more detailed results presented for streptavidin illustrate the ability of the method to exhaustively sample the conformational space of loops presenting several meta-stable conformations.
Availability: We are developing a software package called MoMA (for Molecular Motion Algorithms), which includes modeling tools and algorithms to sample conformations and transition paths of biomolecules, including the application described in this work. The binaries can be provided upon request and a web application will also be implemented in the short future.
Contact: abarozet@laas.fr, jcortes@laas.fr

1 Introduction

Determining structural and dynamic properties of proteins, which are essential to understand their functional roles, remains a challenging goal. Obtaining this valuable information experimentally is expensive and nontrivial. High-resolution structural information can be elucidated via biophysical techniques such as X-ray crystallography and nuclear magnetic resonance (NMR). Given the great value placed on this data, large databases of protein structures have been assembled and organized, for example, in the Protein Data Bank (PDB) (Berman *et al.*, 2000) and the Structural Classification of Proteins (SCOP) (Fox *et al.*, 2014). For the large majority of proteins within these databases, a single structure is recorded. However, proteins are dynamic molecules, some of which exhibit small fluctuations around a stable conformation while others can undergo larger structural rearrangements to participate in critical functions (Boehr *et al.*, 2009). In particular, loop regions may exhibit high flexibility even in very stable proteins (Shehu *et al.*, 2006). This flexibility makes it difficult to investigate protein loops using traditional techniques, resulting in many "solved" protein structures within the PDB and SCOP omitting data for these regions (Petoukhov *et al.*, 2002; Brandt *et al.*, 2008). When loop information is included, it typically represents a single conformation, which does not adequately characterize the (local) structural diversity of the protein (Shehu *et al.*, 2006; Marks *et al.*, 2018).

This paper deals with computational methods for the structural characterization of protein loops, which are an essential complement to experimental techniques. The addressed problem can be formulated as follows: given a protein structure with the loop region omitted, generate an ensemble of feasible loop configurations while leaving the remainder of the protein rigid. Figure 1 illustrates a cartoon representation of a protein. The end points of the loop, shown as red spheres, are treated as anchors. A successfully generated loop connects these two anchors while satisfying a set of structural constraints (correct bond geometry, no atom overlaps, ...). Numerous methods have been proposed over the years to address this problem. Early methods, including ours (Cortés *et al.*, 2004), were mainly based on geometric and numerical approaches (Shenkin *et al.*, 1987; Wedemeyer and Scheraga, 1999; Canutescu and Dunbrack, 2003), whereas most recent methods tend to exploit structural knowledge (Messih

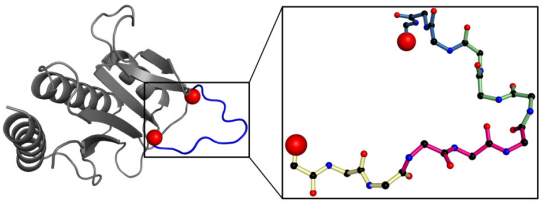


Fig. 1. A loop region of a protein is illustrated between two stationary anchors (spheres). On the right, a more detailed picture highlights the tripeptides of the loop in alternating colors.

et al., 2015; Karami *et al.*, 2018) or a combination of these approaches (Lee *et al.*, 2010; Stein and Kortemme, 2013; Tang *et al.*, 2014; López-Blanco *et al.*, 2016; Marks *et al.*, 2017). Making a concise and meaningful survey of available methods is not an easy task, and goes beyond the aims of this paper. The interested reader can refer to a survey by Shehu and Kavraki (2012).

Many of the aforementioned methods were aimed to predict the most probable conformations of the loop. Thus, sampling is focused on a reduced region of the conformational space. This can be an advantage to efficiently model loops presenting a well-defined and relatively stable structure. However, such loop predictors usually fail when modeling conformationally diverse loops (Marks *et al.*, 2018). More exhaustive sampling is important for an accurate characterization of the conformational energy landscape, enabling the identification of multiple low-energy basins. Moreover, sampling outside the most probable regions is mandatory to investigate loop conformational transitions.

The goal of the method presented in this paper, called MoMA-LoopSampler, is not focused on the prediction of a single structure for a loop, but is aimed at a more thorough exploration of the loop's conformational space. The proposed method employs a hybrid approach to loop modeling that constructs candidate loops utilizing an extensive structural database of small protein fragments and a closed-form inverse kinematics (IK) solver. The loop is divided into a set of consecutive three-residue fragments. All but one of the fragments are iteratively sampled from the database, where each sample is slightly perturbed in order to increase the size of the explored loop configuration space. The last fragment is completed by closing the kinematic chain utilizing an IK solver. MoMA-LoopSampler varies the assignments of which fragment is solved by IK to further increase the sampled space. As each fragment is placed in the candidate loop, collision detection and forward reference checking are performed to prune the search space.

This paper also investigates a more advanced version of the sampler that incorporates a reinforcement-learning-based (RL-based) heuristic. For each fragment within the loop, similar structural pieces are clustered in a low-dimensional projection. Clusters that lead to successfully closed loops are sampled from more frequently. The results illustrate that the RL-based approach preserves loop diversity while improving the performance (number of conformations generated as a function of time) for most of the tested proteins.

In addition, MoMA-LoopSampler proved capable of sampling the two stable states of a loop from the streptavidin protein, as well as intermediate states. Drawing on these results, one can envision the application of MoMA-LoopSampler to investigate the energy landscape of a loop.

The remainder of this paper is organized as follows. Section 2 (*Methods*) describes the details of the MoMA-LoopSampler method. Section 3 (*Results And Discussion*) showcases the ability of both the learning and non-learning approaches to generate a set of diverse loops, and also applies MoMA-LoopSampler to several loop prediction benchmark datasets. A summary of these results along with potential future work are discussed in Section 4 (*Conclusion*).

2 Methods

The MoMA-LoopSampler method operates in two modes. The first mode, referred to as the basic method, utilizes a structural database combined with a technique from robotics to generate loop samples. The second mode builds upon the first by employing a low-dimensional projection to organize the information extracted from the structural database. A RL-based heuristic (H_{RL}) is then utilized to speed up future sampling.

2.1 Protein representation

Proteins are represented using an all-atom model where the degrees of freedom are the backbone dihedral angles ϕ, ψ, ω. Bond lengths and bond angles are held constant as per the idealized model (Engh and Huber, 1991). The loop portion of each protein is decomposed into a set of n tripeptides (continuous segments of a protein comprised of 3 amino acid residues). Each tripeptide is represented by 3 sets of ϕ, ψ, ω angles, referred to as a tripeptide "state" throughout the paper. Loops where the number of residues is not divisible by 3 can also be handled by including additional residues at either side of the loop and by restraining the dihedral angles of these residues. However, for the sake of simplicity, we only explain the case of loops with a number of residues divisible by 3. Sampling of ϕ, ψ, ω angles is performed through the selection of tripeptide states from an appropriate database. Initially, the side-chains of the loop are omitted in the model. They can be added once a closed conformation of the backbone has been found.

2.2 MoMA-LoopSampler without H_{RL}

2.2.1 Loop construction

Algorithm 1 showcases the basic MoMA-LoopSampler. To facilitate using samples from the structural database for each tripeptide in the loop, a set of building plans are constructed (function `ConstructLoopPlans`, line 2). A plan corresponds to one possible order in which the tripeptides of the loop can be assembled. At each iteration, the function `SelectPlan` (line 4) randomly picks a plan that the recursive function `BuildLoopPos` will follow to build a conformation, starting from the tripeptide with index 1 in the plan (line 5). Among the numerous possibilities, our implementation only considers a subset of n possible plans, where n is the number of tripeptides in the loop. Plan number p assembles the loop starting from tripeptide 1 to $p-1$, then from the end of the loop working backwards from tripeptide n to $p+1$. The method attempts to close the loop by utilizing inverse kinematics for the last

tripeptide, p. This technique allows all of the positions within the loop to be sampled from the database.

C contains the working conformation of the protein. Initially, $\mathsf{C}_{\mathsf{init}}$ includes all the atoms for the non-loop portion of the protein. Once a plan is selected in line 4, tripeptides in the loop are recursively assembled employing a backtracking method. For each tripeptide in the loop, the function `SampleTripeptide` utilizes the amino acid sequence of the tripeptide to access the structural database and randomly sample a tripeptide state (line 11). The nine corresponding angles are then slightly perturbed to enable MoMA-LoopSampler to sample the loop conformational space more finely (function `PerturbState`, line 12). Function `InstallTripeptide` (line 13) then installs the sampled and perturbed tripeptide into the working conformation C and checks that it respects all the required constraints (see Section 2.2.3). If it does, the function `BuildLoopPos` is called to continue building a conformation from the next tripeptide in the plan (line 18). When the last tripeptide in the plan is reached (line 16), loop closure is attempted (function `CloseLoop`). If this is successful, the conformation is added to the ensemble Ω (line 30). The method finishes when it reaches an iteration limit or has sampled a pre-defined number of successful loop conformations.

Due to the backtracking search, this method has linear space complexity and exponential time complexity $\mathcal{O}({max_{\text{atts}}}^{n-1})$, where max_{atts} is the maximum number of attempts to be made in each of the $n-1$ tripeptide positions. However, in practice, failures are detected early in the search process, making this a practical approach. Additionally, a timer (not shown in Algorithm 1) limits the total duration of the `BuildLoopPos` procedure to prevent the algorithm from getting trapped.

Algorithm 1: Build Loop

```
1  void BuildLoop(C_init, L_start, L_end)
2    Plans ← ConstructLoopPlans(L_start, L_end)
3    for i ← 1 to Iterations do
4      plan ← SelectPlan(Plans)
5      BuildLoopPos(plan, C_init, 1)
6  bool BuildLoopPos(plan, C, pos_tri)
7    attempts ← 0
8    success ← false
9    while attempts < max_atts and success = false do
10     attempts ← attempts +1
11     tripeptide ← SampleTripeptide(plan, pos_tri)
12     tripeptide ← PerturbState(tripeptide)
13     C', success ← InstallTripeptide(C, tripeptide)
14     if success then
15       if pos_tri = plan.lastIndex then
16         success ← CloseLoop(plan, C')
17       else
18         success ← BuildLoopPos(plan, C', pos_tri +1)
19   return success
20 bool CloseLoop(plan, C)
21   attempts ← 0
22   success ← false
23   while attempts < max_IK and success = false do
24     attempts ← attempts +1
25     tripeptide_IK ← PerturbOmegas(plan, plan.lastIndex)
26     Solutions_IK ← SolveIK(C, tripeptide_IK)
27     foreach sol_IK ∈ Solutions_IK do
28       C', success_sol ← InstallTripeptide(C, sol_IK)
29       if success_sol then
30         Ω ← Ω ∪ C'
31       success ← (success ∨ success_sol)
32     if success then
33       return success
34   return success
```

2.2.2 Tripeptide database

A database of tripeptide states, indexed by their corresponding amino acid sequence, was constructed using the structures of protein domains obtained from SCOP 2.06 (Fox *et al.*, 2014). The 95% ID filtered subset, consisting of PDB-style files for 28,011 domains, was utilized to build the structural database. Secondary structure labels were assigned to each residue using DSSP (Kabsch and Sander, 1983). The construction process is summarized below, with details available in Section S1.2.

Each structure file was processed by passing a sliding window of size 3 along the amino acid sequence. Each resulting tripeptide state was added to the database if none of its residues participate in an alpha-helix or a beta-strand. Domain structures containing NMR data (multiple models) were subject to filtering to eliminate redundancy.

MoMA-LoopSampler constructs loops by concatenating tripeptides, for example t_1 and t_2, which creates a structure with two implicit tripeptides, t_3 and t_4, straddling t_1 and t_2. These implicit states are referred to as synthetic states, as they were not directly extracted from the set of experimentally solved structures. Employing concatenation enables broader sampling of the conformational space (a goal of MoMA-LoopSampler), although not strictly adhering to the statistical distribution of states in the database. An analysis of synthetic states was performed to further justify their use in MoMA-LoopSampler (detailed in Section S1.3).

2.2.3 Tripeptide placement constraints

When a tripeptide is appended to the loop being constructed within C, its acceptance is subject to two constraints. First, a common AI approach known as forward checking is employed to help improve performance (Russell and Norvig, 2009). When the tripeptide database is built, the maximum length of a tripeptide from end to end is recorded for each amino-acid sequence key. If the remaining distance between the two working loop ends cannot be closed by installing the longest tripeptides from the database, the current loop configuration is considered invalid. This enables backtracking as early as possible in the construction process.

To avoid steric clashes in the final structure, the second constraint checks that the installed tripeptide's backbone atoms do not penetrate any of the van der Waals spheres in C. In MoMA-LoopSampler, the C_β atoms are placed simultaneously with the backbone, since their positions are fully determined from the backbone dihedral angles. In order to eliminate conformations that do not leave room for side-chains, the van der Waals radii for the C_β atoms are artificially

increased during the collision detection process. The radii are set depending on the type (and thus size) of the associated side-chains, as originally proposed by Levitt (Levitt, 1976).

Another feature of MoMA-LoopSampler is the ability to add a constraint on the position of an atom. Although this feature is not showcased in the results presented below, it may be interesting if an atom needs to contact another residue, or if its position has been experimentally determined.

2.2.4 Loop closure

The function `CloseLoop` (Algorithm 1) attempts to close the loop by computing the dihedral angles of the last tripeptide (all the other tripeptides are held fixed during this process). This process is detailed in Section S1.4. Briefly, the three ω angles are randomly sampled utilizing a Gaussian distribution centered around 0 or π (function `PerturbOmegas`, line 25). The six remaining ϕ and ψ angles of the backbone are solved via an in-house inverse kinematics (IK) solver (Cortés *et al.*, 2004) (function `SolveIK`, line 26). With 6 degrees of freedom, this problem can be solved in closed-form, and may yield up to 16 solutions. All collision-free solutions are recorded, and the resulting conformations are added to Ω. If no collision free solution is found, the process (ω sampling followed by IK) is repeated until a solution is found, or the maximum number of attempts, max_{IK} is reached. In the latter case, backtracking is employed and the construction continues. Validation of conformations for the closing tripeptides, which are also "synthetic states", and their relationship to loop quality, are detailed in Section S1.5.

2.3 MoMA-LoopSampler with H_{RL}

2.3.1 Objectives and principle

The loop construction method detailed so far effectively discretizes the conformational space of the loop by sampling from the structural database. For small loops (≤ 9 residues), an exhaustive search of all combinations can be completed within a few hours. However, larger loops present a formidable challenge, as computational requirements increase exponentially. In this section, a new method is proposed which utilizes a RL-based strategy (H_{RL}) to improve the naive sampling strategy presented in Algorithm 1. The goal of utilizing H_{RL} for short loops is to provide a more efficient and exhaustive characterization of the loop. For longer loops, especially in highly constrained environments, H_{RL} can quickly prune infeasible areas of the search space, resulting in a more computationally efficient search.

2.3.2 Learning approach

A new approach that incorporates H_{RL} is shown in Algorithm 2. For each loop plan, a learning tree is built (function `ConstructRLTrees`, line 3). This data structure (presented in Section 2.3.3) records statistics about prior tripeptide state selection and their associated participation in successfully closed loops. On line 13, a new function is used to sample a tripeptide state (`SampleTripeptideRL`). This function uses statistics from the appropriate learning tree to guide tripeptide sampling towards zones that have a higher chance of generating successful conformations, or which have not been explored yet. Details about tripeptide state selection can be found in Section 2.3.4.

On line 16, the success or failure of the tripeptide placement is recorded in the learning tree to update the statistics about the tripeptide state's ability to form a successful loop (`RecordSuccessPlacement`). Similarly, on line 22, the closure of the loop (or absence thereof) is recorded (`RecordSuccessClosure`), updating the statistics of all the tripeptide states used in the successfully generated loop.

Algorithm 2: Build Loop RL

```
1  void BuildLoopRL(C_init, L_start, L_end)
2      Plans ← ConstructLoopPlans(L_start, L_end)
3      Trees ← ConstructRLTrees(Plans)
4      for i ← 1 to Iterations do
5          plan ← SelectPlan(Plans)
6          tree ← SelectTree(Trees, plan)
7          BuildLoopPosRL(plan, C_init, 1, tree)
8  bool BuildLoopPosRL(plan, C, pos_tri, tree)
9      attempts ← 0
10     success ← false
11     while attempts < max_atts and success = false do
12         attempts ← attempts +1
13         tripeptide ← SampleTripeptideRL(plan, pos_tri, tree)
14         tripeptide ← PerturbState(tripeptide)
15         C', success ← InstallTripeptide(C, tripeptide)
16         tree ← RecordSuccessPlacement(tripeptide, tree, success)
           if success then
17             if pos_tri = plan.lastIndex then
18                 success ← CloseLoop(plan, C')
19             else
20                 success ← BuildLoopPosRL(plan, C', pos_tri+1,
                   tree)
21     tree ← RecordSuccessClosure(tripeptide, tree,
       success)
22     return success
```

2.3.3 Learning data structure

Tripeptides are projected into a space of low dimension m and organized in a m-dimensional tree data structure (see Section 2.3.5). This structure is called an *octree* when $m = 3$. Herein, this m-dimensional tree data structure will be referred to as a *tree*. For each tripeptide amino-acid sequence, all corresponding states from the database are projected to compute a bounding box $\mathcal{B}$ within the lower-dimensional space. Box $\mathcal{B}$ is divided into 2^m sub-boxes, or *cells*, by passing orthogonal hyperplanes through its center. Each cell can then be subdivided using the same method, creating a tree structure. Final cells (those that are not subdivided) are called *leaves*. Initially, all trees are of depth 1, containing 2^m leaves.

This data structure groups together tripeptide states that are close to one another in the chosen lower-dimensional space. Each leaf holds statistics about the group of tripeptide states it contains, namely their ability to form a closed loop without collisions. A leaf is subdivided (and thus is no longer a leaf) when the statistics about the states within it are too heterogeneous, indicating very different behaviors with respect to loop construction (see Section S1.6). This new subdivision is aimed at separating the tripeptide states into

homogeneous groups with respect to their participation in successfully constructed loops.

The complete data structure uses chained trees. Each loop plan is associated with a chain of m-dimensional trees organized as follows: the first tree (the *root tree*) contains the possible states for the first tripeptide to be built in the plan. Each leaf within the first tree points to another tree containing the possible states for the second tripeptide to be built in the plan, and so on until obtaining the tree for the penultimate tripeptide.

As an example, assume that tripeptides 1 to $k-1$ in the plan have been sampled so far, and that we are sampling a state for tripeptide k. For j from 1 to $k-1$, let us call c_j the state chosen for tripeptide j, and $\mathcal{L}_j$ the leaf that holds the statistics about the state c_j. $\mathcal{L}_1$ is the leaf of the root tree containing c_1. $\mathcal{L}_2$ is the leaf of the tree pointed to by $\mathcal{L}_1$ containing c_2, and so on. The statistics about the sampling and placement of tripeptide k will be recorded in the tree $\mathcal{T}_k$ pointed to by $\mathcal{L}_{k-1}$.

$\mathcal{T}_k$ contains the states available in the database for tripeptide k. This structure stores statistics about the ability of these states to participate in forming a successful loop once the states for tripeptides 1 to $k-1$ have been chosen in leaves $\mathcal{L}_1$ to $\mathcal{L}_{k-1}$, respectively. The statistics collected are: the number of times a state it contains could be placed while respecting all constraints (forward checking and collisions), the number of times it could not be placed, and the number of successfully closed loops containing tripeptide states within this leaf.

2.3.4 Tripeptide selection

When a state is sampled from the database for a given tripeptide, the statistics about the previously sampled states are used to guide the choice. The learning tree corresponding to the current tripeptide is selected given the states of the already placed tripeptides. A score is associated to each leaf of the tree. For the tests presented here, the score S for leaf $\mathcal{L}$ is set as:

$$S = \begin{cases} N \cdot S_{\max} & \text{if at least one state in } \mathcal{L} \text{ has been used to build a successfully closed loop} \\ N \cdot \min(S_{\max}, T) & \text{otherwise, with } T = \max\left(S_{\min}, \frac{t^{n-k}}{a}\right) \end{cases}$$

In this formula, $S_{\min}$ and $S_{\max}$ are parameters setting lower and upper limits on the score, respectively. N is the number of tripeptide states in $\mathcal{L}$, and a is the number of times a state from $\mathcal{L}$ has been sampled. k is the position of the tripeptide in the plan and n is the number of tripeptides in the loop. Finally, t is a positive real number setting the learning rate. Lower values of t lead to higher learning rate (i.e. to a greedier learning process). It is an important parameter as it is determinant for the diversity of the loop ensemble and the speed at which loops will be generated (investigated in Section 3).

When sampling a state for a given tripeptide with `SampleTripeptideRL` (Algorithm 2, line 13), a leaf is randomly picked among all the tree's leaves using the probabilities corresponding to the normalized scores. Then, a state is chosen from the selected leaf using uniform random sampling.

T acts as a threshold score. Ideally, the score of leaves containing no working states (successful loop closures) should be zero. In practice, it is impossible to guarantee that no state in a leaf is able to lead to a successful loop, even when those have all been tested (because of the small perturbations and the fact that the tripeptide states sampled upstream may be different from one attempt to another). Therefore, this method maintains the score so that the leaf has a non-zero chance of being explored even when it has failed to lead to a successful loop closure. After each failed attempt, the threshold score decreases until reaching a lower limit. However, as soon as a state is found that leads to a successful loop, the score is set back to its maximum value, which it maintains for the remainder of the search.

Note that this scoring approach does not involve a cumulative reward, characteristic of standard RL methods, which is why it is referred to as a RL-based heuristic. Other scoring approaches, possibly involving cumulative reward, can be applied within our method. However, among the options we tested, this score is the one that best preserved the diversity among sampled conformations.

2.3.5 Tripeptide projection

Choosing an appropriate tripeptide state projection is of crucial importance for H_{RL} to be effective. The goal of the tripeptide projection is to group similar states together so that we can infer a state's success from the results obtained by its neighbors. If the projection is ill-chosen, it may group states that yield heterogeneous outcomes with respect to loop closure, in which case no correct assumption can be made of one state even if we know the success of a neighboring state within the projection.

Several options were investigated for the tripeptide state projection (full list and empirical experiments shown in Section S1.7). In this work, we selected the projection named *Position*, which is the vector representing the relative position of the N and C atoms at the two ends of the tripeptide. Thus, it is a projection in dimension $m = 3$. Although none of the investigated projections are clearly superior to others for all systems, results show that *Position* is a reasonable choice given its: low dimensionality, uniform distribution of tripeptides, and ability to cluster states with respect to successful loop closure.

3 Results and discussion

We applied our loop modeling framework to a few benchmark sets of proteins. We first present the results obtained without activating H_{RL}, showcasing the ability of basic MoMA-LoopSampler to sample meta-stable conformations as well as transition regions, and its computational efficiency. The performance of H_{RL} is then detailed.

3.1 Tests performed and visualization of results

3.1.1 "Native" loop

In all that follows, the "native" loop is the name given to the conformation in the crystal structure, even though due to the inherent flexibility of protein loops it may not be the only (meta-)stable conformation the loop can adopt. Although the ability to sample the native conformation is by no means evidence that the method is capable of sampling all relevant conformations, any exhaustive sampling must contain this conformation, which is why distance to native is used as a means to estimate the quality of the generated ensemble. We define the RMSD_{min} of a loop ensemble as the lowest backbone RMSD between a loop conformation from the ensemble and the native conformation, after aligning the fixed portion of the protein.

3.1.2 Test sets

The performance of our method was tested on three benchmark sets of 9, 12, and 15-residue loops. The 9 and 12-residue test sets are taken from the loops gathered by Jacobson *et al.*, 2004, and the 15 residue test set contains the 30 15-residue loops used in the analysis by Zhao *et al.*, 2011. The complete list of loops utilized in these tests is available in the Supplementary Material, Section S2.1 and Tables S1 to S3.

3.1.3 Test parameters

In all tests, the max_{atts} parameter was set to 10 for 9-residue loops, 7 for 12-residue loops and 5 for 15-residue loops. The max_{IK} parameter was set to 100, and the maximum time for the `BuildLoopPos` and `BuildLoopPosRL` functions was set to 20 seconds. Two atoms separated by more than 3 bonds were considered in collision if the distance between them was below 0.7 times the sum of their van der Waals radii (Bondi, 1964).

For runs performed with H_{RL}, *very low learning rate* results correspond to the results obtained with parameter t (see Section 2.3.4) set to 15, *low learning rate* results with $t = 10$, *high learning rate* results with $t = 2$, and *very high learning rate* results with $t = 1$.

MoMA-LoopSampler also supports brute force exploration (Section S1.1). It exhaustively explores the conformational space by trying every possible combination of tripeptide states from the database for each possible plan, obtaining the full conformational space that is reachable by MoMA-LoopSampler given the database utilized.

3.1.4 2D Loops projections

A convenient way to visualize the sampled regions in the conformational space is to plot two-dimensional projections based on two meaningful descriptors for each loop. The generated plots can give insight into the density of the sampled conformations in different regions. It is especially convenient to compare the conformational ensembles obtained under two different conditions.

The first dimension, d_1 (x-axis), is the distance (Å) between an atom located in the middle of the loop and a fixed atom in the protein. The second dimension, d_2 (y-axis), is the angle (°) formed by three atoms at approximately one quarter, one half, and three quarters of the loop sequence.

3.2 Results obtained without the RL-based heuristic

3.2.1 Comparison with prediction methods

Although loop prediction is not the primary purpose of MoMA-LoopSampler, the ability to sample conformations found in crystallographic structures is a requirement for any loop sampling method. To assess the ability of MoMA-LoopSampler to sample near-native conformations in reasonable time, we compared its performance in this regard to that of the sampling phase of DiSGro (Tang *et al.*, 2014) and of the updated version of RCD (Chys and Chacón, 2013; López-Blanco *et al.*, 2016) on benchmark sets of loops of length 9, 12 and 15 residues (see Section S2.2.2). Results show that due to the stricter collision constraints enforced by our method, MoMA-LoopSampler generates slightly fewer conformations than other methods given the same computational budget. However, MoMA-LoopSampler manages to more accurately approximate the crystal conformations when the same number of sampled conformations are considered.

Overall, these results suggest that ensembles generated by MoMA-LoopSampler contain a lower proportion of unfavorable conformations compared to ensembles generated by other methods. Generating fewer conformations, but which are more representative of the accessible conformational space, is essential considering the costly downstream processing steps of applications involving loop sampling (e.g. side-chain placement, relaxation, scoring, clustering).

3.2.2 Application to a multi-state loop

We demonstrate the exhaustive sampling ability of MoMA-LoopSampler by generating relevant loop conformations using the streptavidin protein. Streptavidin is a homotetramer protein that strongly binds biotin. Each monomer exhibits a biotin binding site and a flexible loop L (between residues 44 and 52) that stabilizes the complex by "closing" upon binding. Two conformations are known for L: "open" and "closed". Sampling loop conformations from several structures of the protein with MoMA-LoopSampler, we intend to explain the presence of the conformation in the crystal structure, and to determine which of the known conformations are accessible to the loop.

We extracted three high-resolution structures of streptavidin from the PDB: 2F01 (Le Trong *et al.*, 2006), 3RY1 (Le Trong *et al.*, 2011) and 3RY2 (Le Trong *et al.*, 2011). 2F01 and 3RY2 both contain two subunits in the asymmetric unit, whereas 3RY1 contains four subunits. The subunits from 2F01 and 3RY2 are all bound to a ligand (either biotin or epi-biotin), with their L loops in the "closed" conformation. The subunits from 3RY1 are all unbound, however, one of them shows L in the "closed" conformation, while the others have L in the "open" conformation.

We separated all the subunits and used MoMA-LoopSampler to perform a brute force exploration of L's conformational space from the different subunits of the three starting crystallographic structures, after removing the ligand (if present). After side-chain placement, the sampled conformations were relaxed and their energy was estimated using Amber (see Section S2.3.1). They were then projected in 2D space (see Section 3.1.4) and showcased in Figure 2.

The projected conformations adopt an overall triangular shape in which the "closed" conformations are projected in the lower left vertex, while the "open" conformations are more diverse and are projected on the opposite side of the triangle. Combining MoMA-LoopSampler with the energy calculation creates landscapes that are consistent with (and justify) the conformations adopted in the crystals, and can predict how flexible the loop is.

Whichever the employed scaffold, the crystallographic conformation is always found in a low-energy basin. Starting from the protein conformations extracted from crystal structures of the complex, both the "open" and "closed" conformations appear to be in low-energy basins, suggesting that both conformations are accessible to the loop when the ligand is removed. Employing MoMA-LoopSampler allows us not only to identify *both* stable conformations, but also to visualize the topography of the landscape modeled by the energy function, including the transition region connecting the two basins.

The landscape is different when the starting structure for the protein is extracted from a crystal formed without

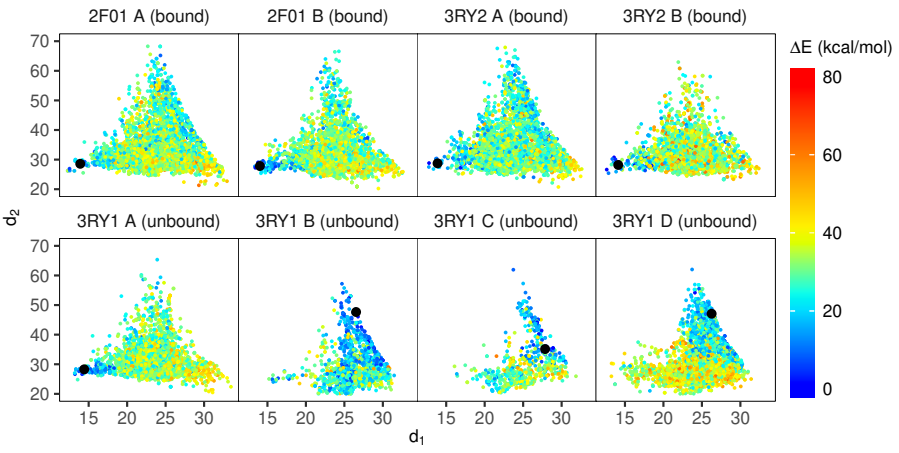


Fig. 2. 2D projections of conformations sampled using MoMA-LoopSampler in brute force mode for a loop in streptavidin, from eight starting X-ray protein structures. The first dimension, d_1 (x-axis), is the distance (Å) between an atom located in the middle of the loop and a fixed atom in the protein. The second dimension, d_2 (y-axis), is the angle (°) formed by three atoms: an atom at approximately one quarter, one half, and three quarters of the way down of the loop. The conformations from the crystallographic structures are shown in black. For each system, the loop with the lowest energy was identified and each conformation was then colored according to the difference between its energy and this lowest energy.

ligand. In the case of 3RY1 (A), the area around the "open" conformation could not be sampled, and only the "closed" conformation is projected into an energy basin. This suggests that the loop environment is more constrained sterically, and that it strongly stabilizes this "closed" conformation. Although a prediction method might have predicted that the loop would adopt this unexpected conformation, employing MoMA-LoopSampler allows us to reconstruct the whole landscape and to confirm that the loop is stabilized in that position.

From the other three scaffolds extracted for the unbound protein, the landscape shows a large basin around the "open" conformation, predicting that the loop will be rather flexible and adopt an "open"-like conformation.

Very similar landscapes were obtained using MoMA-LoopSampler with different learning rates for H_{RL} (Section S2.5). However, landscapes generated using DiSGRo (Tang *et al.*, 2014) and RCD+ (López-Blanco *et al.*, 2016) as sampling methods (Section S2.3.3) present major differences. Results clearly show that the ensembles generated using MoMA-LoopSampler, which are better filtered, allow a much finer analysis of the landscapes.

In addition to analyzing the energy landscape of a loop, MoMA-LoopSampler can be used to sample intermediate states between two stable conformations. For example, in the cases in which the "open" and "closed" basins are both present and of low energy. This could then enable the analysis of the docking mechanism of streptavidin and biotin in great detail. Nevertheless, the analysis of conformational transitions goes beyond the scope of this paper.

3.3 Performance of the RL-based heuristic

Results of MoMA-LoopSampler with H_{RL} are described and compared to results obtained utilizing the basic method. We analyze the benefits of using H_{RL}, as well as the potential downfalls, mainly in terms of loop diversity. Four tests were performed in the basic mode, as well as four tests in H_{RL} mode, using different learning rates (see Section 3.1.3). In each of these overall eight tests, 9-residue loops were sampled for 2 hours, 12-residue loops for 4 hours and 15-residue loops for 6 hours.

3.3.1 Number of conformations sampled

The main interest of H_{RL} is that it enables faster generation of loop conformations, as shown by the higher densities in Figure 3. In the basic mode, around 20,000 conformations for 9-residue loops and 35,000 conformations for 12- and 15-residue loops were generated on average during the allotted sampling time. In H_{RL} mode, (depending on the learning rate), between 26,000 and 38,000 samples of 9-residue loops and between 34,000 and 53,000 samples of 12- and 15-residue loops were generated. The *very high* learning rate generates 127% more conformations on average compared to the basic mode, but this percentage is highly variable across loop systems. Loop 73, which is located in a very constrained environment, constitutes an extreme case in which activating H_{RL} can multiply by over 41 the number of conformations sampled over 6 hours. Loops 68, 21 and 85 are other very successful examples, for which H_{RL} multiplies by 12, 5.4 and 5.2 the number of sampled conformations, respectively. However, for a few loops, H_{RL} was found to actually decrease by up to 12% the number of sampled conformations. This is actually due to the overhead of the learning process itself: each time a loop is sampled, statistics are updated and the learning trees are maintained. In a few cases, the time saved during conformation sampling itself (especially for *very low* or *low* learning rates) is not high-enough to compensate for the time lost in maintaining the learning data structures.

Overall, employing higher learning rates tends to produce a higher number of conformations. However, also due to the overhead of maintaining a very large tree, or to the nature of the sampled loop, there are several exceptions to that trend. These are illustrated and discussed in Section S2.4.1.

3.3.2 Effect of H_{RL} on the quality of the ensemble

The RMSD_{min} per loop is shown in Figure 4 for the different learning rates and for the sampling performed without H_{RL}. H_{RL} runs generate conformations as close to native as runs utilizing the basic mode (or even closer) for 9 and 12-residue loops. However, the effect of on longer loops is more ambiguous.

The effect of H_{RL} on the time needed to sample close to the native loop is analyzed in Section S2.4.2 of Supplementary Material. Briefly, results show that H_{RL} modifies the probability to sample tripeptides compared to the basic mode, possibly resulting in slightly longer times to approximate the native loop. However, this also suggests that using appropriate weights for the different cells of the learning tree directly

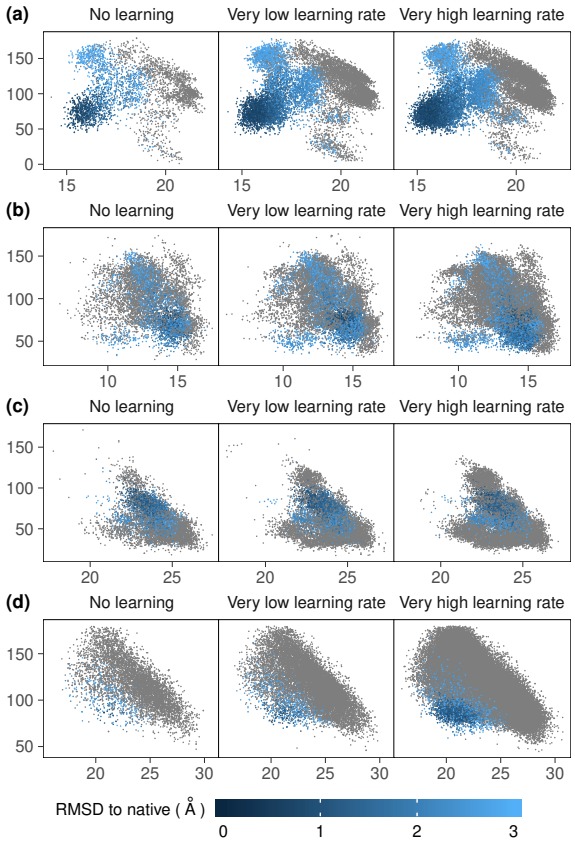


Fig. 3. 2D projections of the sampled loops at different learning rates. Each conformation is represented by a point colored according to its RMSD to native, or in grey if this distance is above 3 Å. The first dimension (x-axis), is the distance (Å) between an atom located in the middle of the loop and a fixed atom in the protein. The second dimension (y-axis), is the angle (°) formed by three atoms: an atom at approximately one quarter, one half, and three quarters of the way down of the loop. (a) Loop 21, (b) Loop 26, (c) Loop 40, (d) Loop 68.

influences the ensemble generated, which may be desirable if one needs to sample with certain distribution guarantees.

As the sampling progresses, the method learns which cells in the tree have not led to successful loop conformations so far, and progressively reduces the probability of sampling inside them. If the algorithm learns too fast, this may happen even though the states explored in the cell are not adequately representative. Therefore, a careful parameterization of H_{RL} is crucial to get an exhaustive sampling. We explored the diversity of the ensembles sampled with different learning rates in order to determine if all areas of conformational space are adequately covered. Figure 3 shows the 2D projections of the loop samples obtained by employing various learning rates on four different systems. The most obvious observation is that all projections corresponding to the same system look similar, in the sense that they have the same overall shape. Even with a *very high* learning rate, there does not seem to be major areas of conformational space that are ignored. However, some sparse areas of the 2D projection space may no longer be sampled. For example, in Figure 3(c), the area at the top left-hand corner is void of conformations with the *very high* learning rate. The same observation can be made for the top-most region of Figure 3(b).

A very striking observation is that the 2D projection plots are denser at higher learning rates. This is a natural result of sampling more conformations in the same amount of time. However, the distribution of points within these projections indicates that in H_{RL} mode, MoMA-LoopSampler samples the conformational space with a higher resolution than in the basic mode. H_{RL} provides greater diversity in sampled areas. When the area around the native loop gets explored more densely, the native loop can be found with better accuracy. This is clearly the case for loops 21 and 68 (Figures 3(a)(d) and 4).

Looking at the evolution of sampling as a function of time (Section S2.4.4), two effects of H_{RL} come to light. The first one is that the number of sampled conformations increases, and so does the coverage of the conformational space. The second is that some regions of the space suddenly get "unlocked". This is particularly clear for the *very high* learning rate, where some areas are first ignored due to poor results in the region, but sampling in the region becomes probable again after one successful conformation is found in the vicinity.

The results provided here suggest that H_{RL} allows for a much faster conformational sampling, although the generated ensembles may lose diversity if the learning process is too greedy. It is therefore of crucial importance to limit learning rates to preserve the diversity enabled by the tripeptide database.

4 Conclusion

This paper has introduced MoMA-LoopSampler, a new method that employs local sequence-dependent structural knowledge and geometric techniques combined with a reinforcement-learning-based heuristic to exhaustively and efficiently sample protein loop conformations. The results show that this new method performs similarly to (or better than) existing computational methods in terms of computational efficiency and that the ensemble of sampled loop conformations includes those found in experimental structures (the "native" state of the loop). The implemented reinforcement-learning-based approach allows MoMA-LoopSampler to accelerate sampling while maintaining conformational diversity (avoiding "over-learning"), and is scalable to large loop regions (15 residues). This work has also shown that MoMA-LoopSampler enables modeling loops present in several low-energy basins, thus being a useful tool when investigating energy landscapes and studying conformational transitions.

Further enhancements to the method include improving the learning component to limit its memory requirements. Another area to investigate is adjusting the scores of leaves within the learning structure so that the distribution of sampled conformations corresponds to the distribution of tripeptide states present in the database. Database adjustments may also improve the quality of the results. For example, filtering the database to only keep one representative among very similar tripeptides would speed up the sampling, and adding the states of similar sequences to the states of rare tripeptide sequences may allow the sampling of conformations currently inaccessible due to a potential lack of data.

Finally, building relevant loop ensembles requires both a sampling and a scoring components. While MoMA-LoopSampler is aimed at providing a diverse ensemble of possible conformations, it does not evaluate the sampled

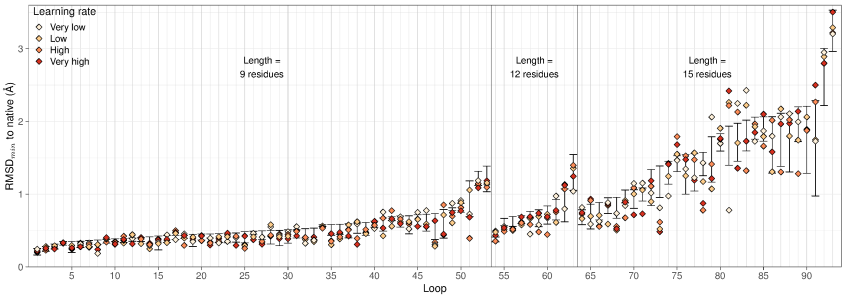


Fig. 4. RMSD$_{min}$ of MoMA-LoopSampler on each loop ensemble. Diamonds correspond to the results obtained for each learning rate. Error bars show the RMSD$_{min}$ range obtained by the four tests performed without H_{RL}. A data point above the error bar shows that the corresponding run with H_{RL} misses some closer-to-native conformations that can be sampled when turning H_{RL} off. A data point below the error bar indicates that the corresponding run found conformations that are even closer to native than the runs without learning did.

loops or estimate their likelihood. Designing an appropriate scoring function, or integrating existing ones into MoMA-LoopSampler, constitutes an interesting direction for future work.

Funding

The French National Association of Research and Technology (ANRT) is gratefully acknowledged for supporting A.B. (contract 2016/0239). This work used the HPC resources of the CALMIP supercomputing center (allocation 2016-P16032).

5 References

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., and Bourne, P. E. (2000). The Protein Data Bank. *Nucleic Acids Res.*, **28**(1), 235–242.

Boehr, D. D., Nussinov, R., and Wright, P. E. (2009). The role of dynamic conformational ensembles in biomolecular recognition. *Nat. Chem. Biol.*, **5**(11), 789–796.

Bondi, A. (1964). van der Waals Volumes and Radii. *J. Phys. Chem.*, **68**(3), 441–451.

Brandt, B. W., Heringa, J., and Leunissen, J. A. M. (2008). SEQATOMS: a web tool for identifying missing regions in pdb in sequence context. *Nucleic Acids Res.*, **36**(suppl_2), W255–W259.

Canutescu, A. A. and Dunbrack, R. L. (2003). Cyclic coordinate descent: A robotics algorithm for protein loop closure. *Protein Sci.*, **12**(5), 963–972.

Chys, P. and Chacón, P. (2013). Random Coordinate Descent with Spinor-matrices and Geometric Filters for Efficient Loop Closure. *J. Chem. Theory Comput.*, **9**(3), 1821–1829.

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

Engh, R. A. and Huber, R. (1991). Accurate bond and angle parameters for X-ray protein structure refinement. *Acta Crystallogr., Sect. A.: Found. Adv.*, **47**(4), 392–400.

Fox, N. K., Brenner, S. E., and Chandonia, J.-M. (2014). SCOPe: Structural Classification of Proteins–extended, integrating SCOP and ASTRAL data and classification of new structures. *Nucleic Acids Res.*, **42**(D1), D304–D309.

Jacobson, M. P., Pincus, D. L., Rapp, C. S., Day, T. J. F., Honig, B., Shaw, D. E., and Friesner, R. A. (2004). A hierarchical approach to all-atom protein loop prediction. *Proteins*, **55**(2), 351–367.

Kabsch, W. and Sander, C. (1983). Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, **22**(12), 2577–2637.

Karami, Y., Guyon, F., De Vries, S., and Tufféry, P. (2018). Dareus-loop: Accurate loop modeling using fragments from remote or unrelated proteins. *Sci. Rep.*, **8**(1), 13673.

Le Trong, I., Aubert, D. G. L., Thomas, N. R., and Stenkamp, R. E. (2006). The high-resolution structure of (+)-epi-biotin bound to streptavidin. *Acta Crystallogr., Sect. D: Biol. Crystallogr.*, **62**(Pt 6), 576–581.

Le Trong, I., Wang, Z., Hyre, D. E., Lybrand, T. P., Stayton, P. S., and Stenkamp, R. E. (2011). Streptavidin and its biotin complex at atomic resolution. *Acta Crystallogr., Sect. D: Biol. Crystallogr.*, **67**(Pt 9), 813–821.

Lee, J., Lee, D., Park, H., Coutsias, E. A., and Seok, C. (2010). Protein Loop Modeling by Using Fragment Assembly and Analytical Loop Closure. *Proteins*, **78**(16), 3428–3436.

Levitt, M. (1976). A simplified representation of protein conformations for rapid simulation of protein folding. *J. Mol. Biol.*, **104**(1), 59–107.

López-Blanco, J. R., Canosa-Valls, A. J., Li, Y., and Chacón, P. (2016). RCD+: Fast loop modeling server. *Nucleic Acids Res.*, **44**(W1), W395–W400.

Marks, C., Nowak, J., Klostermann, S., Georges, G., Dunbar, J., Shi, J., Kelm, S., and Deane, C. M. (2017). Sphinx: Merging knowledge-based and ab initio approaches to improve protein loop prediction. *Bioinformatics*, **33**(9), 1346–1353.

Marks, C., Shi, J., Deane, C. M., and Valencia, A. (2018). Predicting loop conformational ensembles. *Bioinformatics*, **34**(6), 949–956.

Messih, M. A., Lepore, R., and Tramontano, A. (2015). LoopIng: a Template-Based Tool for Predicting the Structure of Protein Loops. *Bioinformatics*, **31**(23), 3767–3772.

Petoukhov, M. V., Eady, N. A. J., Brown, K. A., and Svergun, D. I. (2002). Addition of missing loops and domains to protein models by x-ray solution scattering. *Biophys. J.*, **83**(6), 3113–3125.

Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, 3rd edition.

Shehu, A. and Kavraki, L. E. (2012). Modeling Structures and Motions of Loops in Protein Molecules. *Entropy*, **14**(12), 252–290.

Shehu, A., Clementi, C., and Kavraki, L. E. (2006). Modeling protein conformational ensembles: From missing loops to equilibrium fluctuations. *Proteins*, **65**(1), 164–179.

Shenkin, P. S., Yarmush, D. L., Fine, R. M., Wang, H. J., and Levinthal, C. (1987). Predicting antibody hypervariable loop conformation. I. Ensembles of random conformations for ringlike structures. *Biopolymers*, **26**(12), 2053–2085.

Stein, A. and Kortemme, T. (2013). Improvements to robotics-inspired conformational sampling in rosetta. *PLoS One*, **8**(5), e63090.

Tang, K., Zhang, J., and Liang, J. (2014). Fast Protein Loop Sampling and Structure Prediction Using Distance-Guided Sequential Chain-Growth Monte Carlo Method. *PLoS Comput. Biol.*, **10**(4), e1003539.

Wedemeyer, W. J. and Scheraga, H. A. (1999). Exact analytical loop closure in proteins using polynomial equations. *J. Comput. Chem.*, **20**(8), 819–844.

Zhao, S., Zhu, K., Li, J., and Friesner, R. A. (2011). Progress in Super Long Loop Prediction. *Proteins*, **79**(10), 2920–2935.

Supplementary Material

A Reinforcement-Learning-Based Approach to Enhance Exhaustive Protein Loop Sampling

Amélie Barozet [1,2,*], Kevin Molloy [3], Marc Vaisset [1], Thierry Siméon [1] and Juan Cortés [1,*]

[1]LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
[2]Sanofi recherche & développement, Integrated Drug Discovery, Molecular Design Sciences, 13 quai Jules Guesde, BP 14, 94403 Vitry-sur-Seine Cedex, France
[3]James Madison University, Harrisonburg, Virginia, USA

*To whom correspondence should be addressed.

S1 Methods: details and analysis

S1.1 MoMA-LoopSampler in Brute Force mode

Algorithm S1.1 shows the Brute Force version of MoMA-LoopSampler. When the conformational space of the loop is restricted, it captures the full conformational ensemble that can be generated by MoMA-LoopSampler.

Since the construction method perturbs the sampled tripeptides, every brute force search in our tests is carried out three times to obtain a set of loop conformations that is representative of the conformational space reachable by MoMA-LoopSampler. The reinforcement-learning-based heuristic (H_{RL}) can also be used to obtain the final learning tree and analyze the distribution of the solutions.

Algorithm S1.1: Build Loop Brute Force

```
1  BuildAllLoops(C_init, L_start, L_end)
2    Plans ← ConstructLoopPlans(L_start, L_end)
3    Trees ← ConstructRLTrees(Plans)
4    foreach plan ∈ Plans do
5      tree ← SelectTree(Trees, plan)
6      BuildLoopsPosBF(plan, tree, C_init, 1)
7  int BuildLoopsPosBF(plan, tree, C, pos_tri)
8    nb_sols ← 0
9    Tripeptides ← GetAllTripeptidesStates(plan, pos_tri)
10   foreach tripeptide ∈ Tripeptides do
11     tripeptide ← PerturbState(tripeptide)
12     C', success ← InstallTripeptide(C, tripeptide)
13     tree ← RecordSuccessPlacement(tripeptide, tree, success)
       if success then
14       if pos_tri = plan.lastIndex then
15         success ← CloseLoop(plan, C')
16         tree ← RecordSuccessClosure(tripeptide, tree, success)
17       else
18         nb_closed ← BuildLoopsPosBF(plan, C', pos_tri +1)
19         nb_sols ← nb_sols + nb_closed
20         tree ← RecordSuccessClosureNb(tripeptide, tree, nb_closed)
21   return nb_sols
```

S1.2 Tripeptide database construction

A database of tripeptide states was constructed using the structures of protein domains obtained from SCOP 2.06 (Fox *et al.*, 2014). This collection contains 244,326 domains, extracted from 77,439 PDB entries. The 95% ID filtered subset of the domains, consisting of PDB-style files for 28,011 domains, was utilized to build the structural database. DSSP (Kabsch and Sander, 1983) is employed to assign secondary structure labels to each residue in these files.

Each structure file was processed by passing a sliding window of size 3 along the amino acid sequence. Each resulting tripeptide was added to the database if all of its 3 residues have a DSSP code of T, S, B, G or no code (which corresponds to an unclassified structural type). In other words, no portion of the tripeptide participates in an alpha-helix or beta-strand. The tripeptide state, which corresponds to its 9 backbone dihedral angles (3 sets of ϕ, ψ, and ω) was recorded in the database and indexed by its corresponding amino acid sequence.

A slightly different treatment was applied when the provided domain structure file originated from NMR data. For each structural file that contained more than one model, a distance filter was applied to corresponding tripeptides in each model to avoid redundancy in the database. A tripeptide state was considered sufficiently distant from another tripeptide state, and was thus added to the database, if it met at least one of the two following criteria: the RMSD on ω, ϕ and ψ angles is above 0.3 radians, or one of the nine dihedral angles differs by more than 1 radian.

S1.3 Tripeptide database analysis

S1.3.1 Motivation

Employing a database of protein configurations to discretize the conformational space capitalizes on the prior knowledge that the backbone dihedral angles only occupy a limited range of values, which are dependent on their neighboring amino acids. The technique of sampling from databases has been utilized in many structural biology problems, including loop sampling and de novo structure prediction.

While the success of these applications may show this approach has merit, we extend this idea by validating the

resulting tripeptide states that are formed by joining two tripeptides into a structure. Note that these tripeptides may be concatenated in one direction or in another, so that both directions have to be tested. Figure S1 shows two tripeptide sequences (MVK and PGT) that have been extracted from our database and joined together to construct a larger protein structure. The red lines highlight new tripeptides that are formed from the overlaps. We refer to the states of these tripeptides as synthetic states, since they were not sampled from the database, and thus, their structural validity is unknown.

We propose validating these synthetic states to strengthen the theoretical basis for our proposed approach. This validation occurs as follows. For each pair (i, j) of tripeptides in the database, we extract 4 synthetic states (as shown in Figure S1), formed by the concatenations ij and ji. Next, for each synthetic states, we search the database using the resulting amino acid keys. For the example in Figure S1, these keys would be VKP, KPG, GTM, and TMV respectively. If we are able to locate a state in the database that is similar, we label the synthetic state as valid. Given that our database is built from a small subset of the protein universe, we can not say anything about synthetic states for which we do not find a similar neighbor.

In this work, similarity is measured as the root mean square deviation (RMSD) of the 3 sets of ϕ, ψ and ω backbone dihedral angles that define each tripeptide state. Similarity is established when the RMSD is less than some threshold ϵ.

Fig. S1. Synthetic tripeptide states creation by sampling two states from the database. The four red lines represent the 4 synthetic tripeptide states extracted.

S1.3.2 Results

The database validation analysis was performed with a threshold of 0.5 radians for the dihedral RMSD. To speed up the process, only 1 out of 1,000 synthetic states were randomly selected and tested for a close neighbor. This represents around 9.5 billion synthetic states tested. On average, 85 % of synthetic states had a close neighbor in the database. Figure S2 gives more precise results by sequence. Unsurprisingly for a database of tripeptides involved in coils, sequences containing glycines are the most populated. However, sequences containing rarer amino acids like cysteines, histidines, methionines and tryptophans contain much fewer states. The distribution of synthetic states with a close neighbor in the database shows that sequences containing these amino acids are also the ones for which the proportion of synthetic tripeptides with a close neighbor in the database are the lowest. This may point to a lack of data for these relatively rare sequences. This fact is also supported by Figure S2(d), in which sequences with few representatives in the database are also the ones for which the average distance of synthetic states to the database is the highest.

Cases in which the absence of close neighbors for a synthetic state is not due to lack of data, but to their very low probability to exist in physiological conditions, do not constitute an important issue if one wants to exhaustively sample the conformational space. Generating loops with such states does not prevent any acceptable conformations from being sampled. Conversely, it might be an issue if one's goal is to preserve the distribution of structural preferences encoded in the database when generating a loop ensemble. Nevertheless, generating statistically-meaningful conformational ensembles goes beyond the scope of this paper.

This analysis contributes to the validation of the method consisting in concatenating the tripeptides to build the loop. Indeed, most synthetic states without a close neighbor in the database actually coincide with a rare tripeptide sequence for which data is insufficient. However, the results reveal a limitation of the method, which is the availability of experimental data for rare sequences. Future improvements of MoMA-LoopSampler will therefore include enriching the database with additional states, potentially sharing states across similar sequences.

S1.4 Loop closure

In this section, we provide details for the method `CloseLoop` (Algorithm 1, line 20).

`PerturbOmegas` (line 25) samples the three ω angles by first selecting their *cis* or *trans* configuration. The probability to generate a *cis* ω angle is taken as the frequency of this event in the database for the corresponding residue. According to the selected configuration, the angle is then sampled around the value 0 or π with a Gaussian distribution whose standard deviation follows that observed in the database for this type of residue.

An in-house inverse kinematics (IK) solver (Cortés *et al.*, 2004) is used to solve the six remaining ϕ and ψ angles of the backbone. However, the design of MoMA-LoopSampler allows for other inverse kinematics solvers to be employed (e.g. Manocha and Canny, 1994; Dinner, 2000; Coutsias *et al.*, 2004). Solving this problem may yield up to 16 potential solutions.

The IK solution yields tripeptide states that may not exist within the structural database. Section S1.5 describes a study showing that for the loops with a very low energy after relaxation, the IK solved tripeptide generally has a close structural neighbor within the database (Figure S3). This analysis also shows that upon loop relaxation, the distance of the final tripeptide to the database tends to lower, suggesting that the final tripeptide acts as a buffer that "absorbs" the rigidity of the other tripeptides by being more "lenient" on the distance to the database, and thus more flexible (Figure S4). Therefore, setting a threshold on the distance of the final tripeptide to the database may lead to the generation of higher quality loops, but care has to be taken in order not to be too restrictive on accepted loop conformations.

S1.5 IK-solved tripeptide: distance to database and loop quality

The last tripeptide used to close the loop is the only tripeptide of the generated conformation that has not been sampled from the database. We define the distance between this tripeptide and another tripeptide state in the database as the angular root-mean-squared deviation of their nine backbone dihedral angles. The distance of the last tripeptide to the database is defined as the lowest distance between this tripeptide and

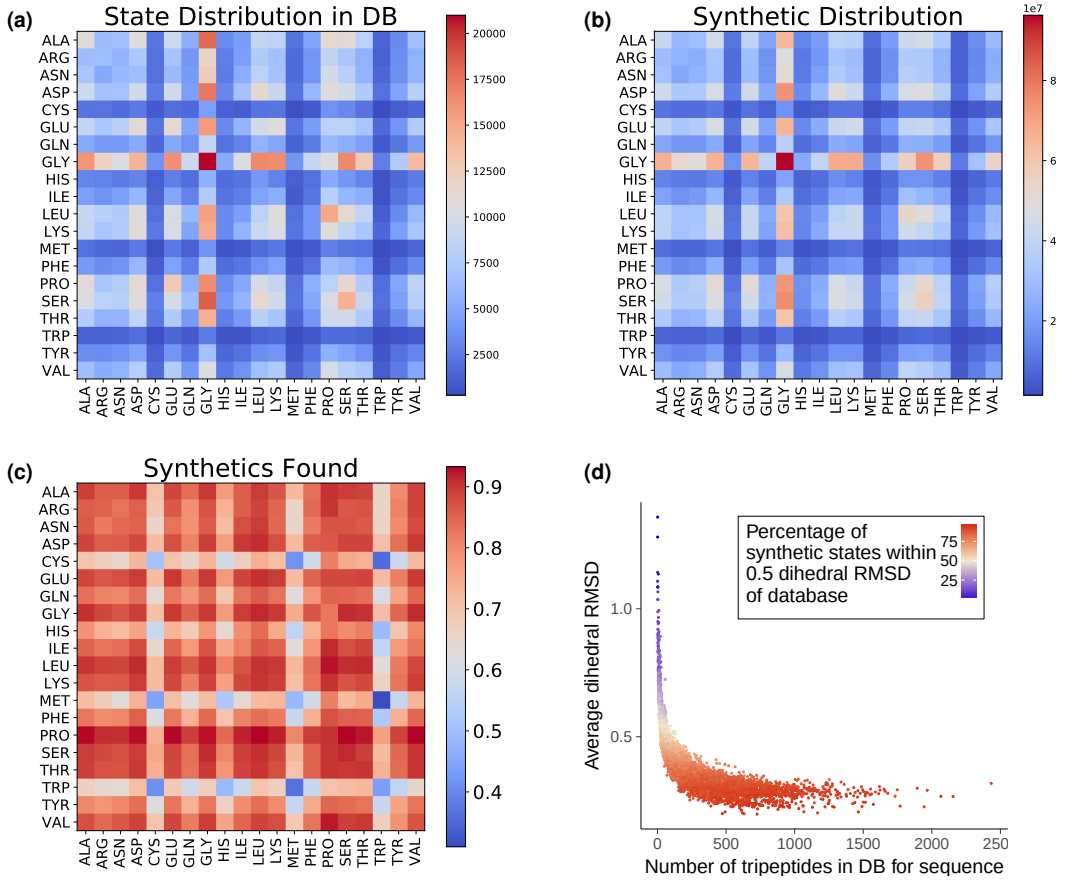


Fig. S2. Each tripeptide state in the database in projected into a 20 by 20 map based on its first two amino acid residues. (a) The top left heatmap shows the distribution of tripeptides using this projection for the SCOP 95% similarity database filtered to exclude states that participate within a secondary structure element. This database contains 2.2 million states. (b) The top right plot is the distribution of synthetic states created by concatenating tripeptide pairs in the database (approximately 9.5 billion states). (c) The bottom left plot shows for each of the synthetic states created, what percentage of these had close structural neighbors in the original database (neighbor distance < 0.5 dihedral RMSD). (d) The bottom right plot shows for each tripeptide sequence the average dihedral RMSD to database obtained for the tested synthetic states depending on the number of actual states in the database.

a tripeptide in the database having the same amino-acid sequence.

For two loop systems, we performed brute force searches of loop conformations with side-chain placement. We then measured the distance to the database for the closing tripeptide of each sampled conformation and recorded the energy of each loop using Amber. The generated loops were then relaxed using Amber and their energy after relaxation was measured. The distance to database of the relaxed closing tripeptide was calculated again.

Figure S3 shows the relationship between the energy of generated loops and the distance of their closing tripeptide to the database. We see clear positive correlation between energy and the distance of the closing tripeptide to the database before relaxation. After relaxation, the profile changes a lot. Although the correlation is still present, it is mainly apparent for low energy loops. Setting a threshold on the distance to database for the closing tripeptide would thus have to be done very carefully.

Figure S4 shows the distribution of the distance to database differences after minus before relaxation for the closing tripeptide. Results show that the distribution is skewed to negative values, indicating a tendency to lower the distance of the closing tripeptide to the database upon relaxation. This suggests that, by being unconstrained with regard to the database, the closing tripeptide acts as a buffer that absorbs the rigidity of the other tripeptides. Relaxation causes the other tripeptides to relax and the closing tripeptide to move closer to states in the database.

Setting a threshold on the distance of the closing tripeptide to the database should be done with a high tolerance since even loops with closing tripeptides far from the database may have a low energy. Moreover, looking at Figure S3(d), we can see that even after relaxation, a loop with a closing tripeptide at distance 1 radian of any tripeptide in the database may be relatively good compared to the generated loop ensemble.

S1.6 Learning mode: leaf subdivision

A leaf in the learning structure can be split if the results obtained for the tripeptide states it contains become too heterogeneous with regard to construction success. The states contained in a leaf should be similar enough to have comparable levels of placement and subsequent loop closing success. When the results show otherwise, the leaf is split around its center. Since each placement event (successful

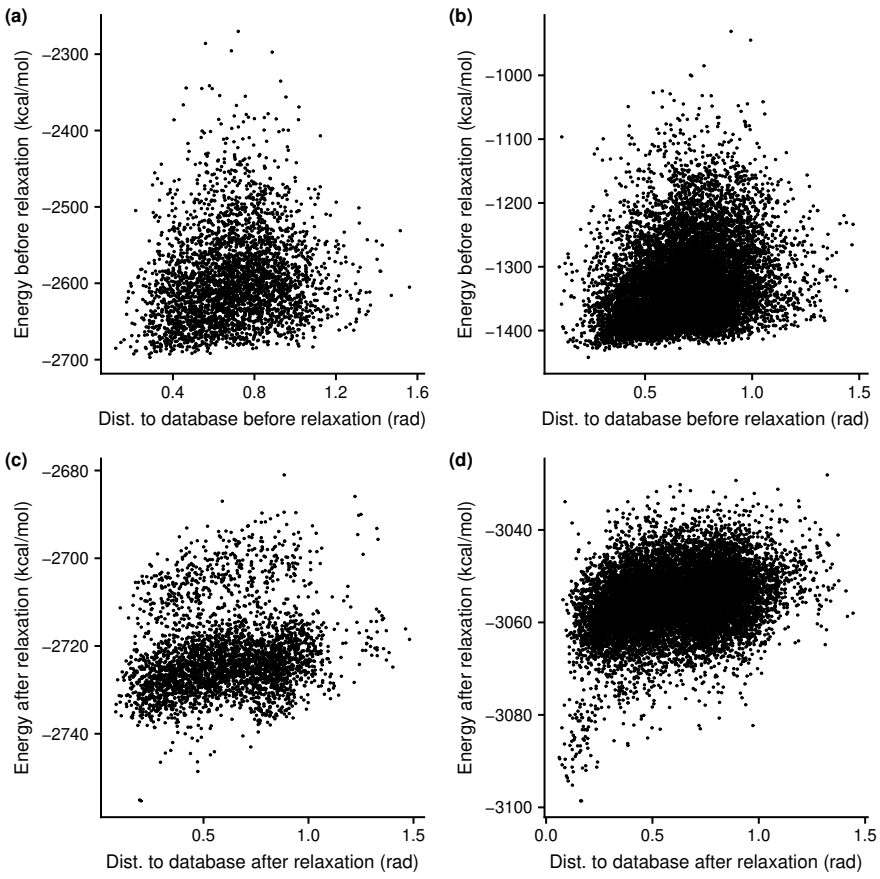


Fig. S3. Amber energy vs. distance to database for the closing tripeptide. (a) and (c) are the graphs for loop 45 (9 residues) while graphs (b) and (d) are the graphs for loop 31 (9 residues). (a) and (b) show the energy and distance before relaxation while (c) and (d) show the energy and distance to database after relaxation.

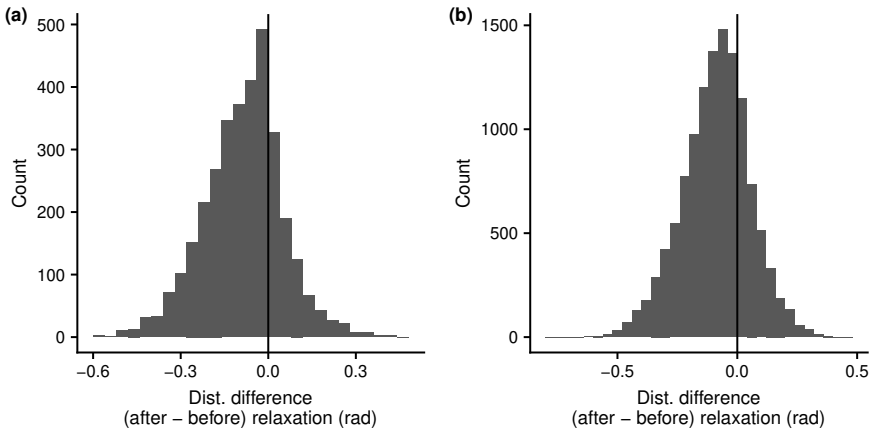


Fig. S4. Histogram of distance to database difference after and before relaxation. Negative values indicate that the distance to database lowered upon relaxation. (a) Loop 45. (b) Loop 31.

placement of a tripeptide, steric clash, loop closure) is recorded along with its corresponding tripeptide state, it is easy to recompute the statistics in each of the 2^m newly created child leaves in case the leaf splits.

Each child leaf obtained after a split is assigned a new tree corresponding to the next position in the plan. Consequently, the statistics from the formerly pointed to tree are also distributed among the newly created trees and their leaves. After this process, the scores of all newly created leaves obtained after the subdivision are the same as what would have been obtained if the tree had utilized this structure from the beginning of the sampling process.

Typically, a leaf split can happen when a state leads to a steric clash between 25% and 75% of the time. Splitting can

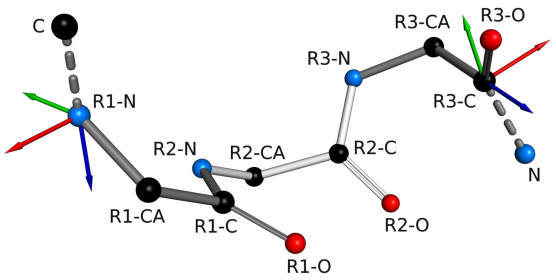


Fig. S5. Frames attached to the beginning and the end of a tripeptide. R1, R2 and R3 designate the first, second and third residues of the tripeptide, respectively. The x-, y- and z- axes are represented in red, green and blue, respectively. Only backbone atoms are represented for clarity. The C atom of the preceding residue and the N atom of the following residue are also visible.

help isolate the states that are responsible for steric clashes, while regrouping the states that lead to a successful placement. Another criterion for splitting the leaf is based on the frequency of forming a closed loop.

S1.7 Learning mode: comparison of tripeptide projections

S1.7.1 Principle

This analysis is aimed at comparing options for tripeptide state projections in the learning process. These options are compared on their ability to evenly distribute states in space and to cluster tripeptides states according to their success in forming closed loops.

S1.7.2 Definitions

The definitions of the different tripeptide state projections are based on the tripeptide geometry corresponding to the nine dihedral angles constituting the state. We associate a reference frame to the beginning and the end of each tripeptide. The frame associated to the beginning of a tripeptide is centered on the N atom of the first residue, while the frame associated to the end is centered on the C atom of the last residue (see Figure S5). The length of a tripeptide is defined as the distance between the first and the last atom of the tripeptide backbone.

Several options were tested for tripeptide state projection. They mostly involve the relative position or orientation of the two ends of the tripeptide. For the orientation, several representations were tested. More precisely, the tested tripeptide state projections are (with the associated dimension m in parentheses):

Position ($m = 3$): The vector of the translational part of the transformation between the beginning and the end of the tripeptide (as defined by the associated frames).

Euler angles ($m = 3$): The vector of the Euler angles of the rotational part of the transformation between the beginning and the end of the tripeptide.

Euler angles and length ($m = 4$): The vector containing the three Euler angles of the rotational part of the transformation between the beginning and the end of the tripeptide and the length of the tripeptide.

Quaternion ($m = 4$): The quaternion representing the rotational part of the transformation between the beginning and the end of the tripeptide.

Quaternion and length ($m = 5$): The vector containing the quaternion representing the rotational part of the transformation between the beginning and the end of the tripeptide, and the length of the tripeptide.

Axis-angle ($m = 4$): The axis-angle representation of the rotational part of the transformation between the beginning and the end of the tripeptide.

Axis-angle and length ($m = 5$): The axis-angle representation of the rotational part of the transformation between the beginning and the end of the tripeptide, and the length of the tripeptide.

S1.7.3 Performance of the different projections

A good projection should provide a good distribution of the tripeptide states in space, and more importantly group together tripeptide states that are almost interchangeable in the loop building process. For example, if two states are very close to one another in the projection space and one is in a collision when placed, then the other should have a strong likelihood to be in collision as well.

A high value of the dimension m would result in a data structure that has very high memory requirements. This would likely result in leaves with too few tripeptides, which is not desirable. Indeed, when a tripeptide state is sampled, its results are exploited to make predictions about the success of all its neighbors in the leaf. If the neighbors are not numerous, H_{RL} will not be very effective.

We ran brute force searches for six loop systems using each of the different projections. We compare the first level of the octrees in all the runs performed, in terms of distribution of tripeptides and success probability of each cell. The first level of the octrees corresponds to the cells obtained after dividing the bounding box containing all tripeptide states once in each dimension. The number of cells at this level is thus 2^m where m is the dimension of the projection. Our method employs several loop construction plans for each loop system (one per tripeptide in the loop, corresponding to a plan ending with this tripeptide). There are therefore 3 plans for the 9-residue loops, and 4 plans for 12-residue loops.

The success probability of a cell is defined as the number of successful combinations of tripeptides using a state from this cell, divided by the theoretical number of tripeptide combinations that use a state from this cell. Ideally, a good projection should distribute the tripeptides as uniformly as possible among the cells, while clearly separating states that work (i.e. that can be used to build a successful loop conformation) from those that cannot lead to a successful loop conformation. Another criterion to consider is the dimension of the projection. A projection with a higher dimension will have higher memory requirements.

Figure S17 (at the end of this document) shows the results for the first levels of all loop systems, for each of their loop construction plan employed. Looking at the different heatmaps, it is clear that no projection performs consistently better than all others. However, we will try to analyze the differences in the results. First, looking at the 3-dimensional projections *position* and *Euler angles*: tripeptide distribution

seems to be rather good in both cases, with very few empty cells. However, the distribution of solutions is more heterogeneous with *position*. Indeed, with this projection, more of the first level cells are void of working solutions. This is particularly striking for 1dim-12 (A213-A224) (loop 55), for the three last plans. *Euler angles* is not able to gather the solutions into only a few cells, whereas *position* concentrates the solutions into two or three cells. The same observation can be made for the other systems. *Position* therefore seems to be a better predictor of tripeptide success in building a loop than *Euler angles*. Comparing the 4-dimensional projections is more delicate. Indeed, *axis-angle*, *Euler angles and length* and *quaternion* behave differently between the different systems. It seems that *Euler angles and length* better gathers the solutions for 1dim-12 (A213-A224) (loop 55) than other projections, but on 153l-12 (A98-A109) (loop 61), *quaternion* is the one that better gathers solutions, while *axis-angle* gathers solutions relatively well in all the cases. For the distribution of tripeptides, the same observation can be made. The quality of the distribution differs depending on the tripeptide sequence. However, 4-dimensional projections do not seem to perform better than *position*. For most systems and most loop plans, there are more than 8 cells left with solutions. Of course those cells contain fewer tripeptide states on average, which is why comparing 3-dimensional and 4-dimensional projections is delicate. In 5-dimensional projections, the distribution of tripeptides results in an undesirable sparsity, where the solutions are unsurprisingly found where tripeptides are located. Therefore these do not stand out from all the projections either.

This comparison is only possible on first level cells with these plots. Many other levels are left to explore and each projection is likely to better separate working and non working states in lower levels. With our analysis, we decided to retain a 3-dimensional projection in order to limit the size of the learning tree in memory. In light of the results, we thus selected *position* as the tripeptide projection for our tests.

S2 Results

S2.1 Loop systems

The performance of MoMA-LoopSampler was tested on three benchmark sets of 9-residue loops, 12-residue loops and 15-residue loops. The 9-residue test set is a subset of the loops gathered by Jacobson and colleagues (Jacobson *et al.*, 2004). 2alp-139 and 8ruc-79 were removed because they are not included in the set by Soto and co-workers (Soto *et al.*, 2008). 1ivd-244 and 1pda-108, which were excluded from the modified Fiser set by DePristo and colleagues for either poor quality or missing side-chain atoms and gaps, were also removed (DePristo *et al.*, 2003). Finally, 4gcr-94 was removed because it only contains 3 turn residues surrounded by β-sheet and α-helix residues, and is not strictly speaking a loop. Thus, the 9-residue test set involves 53 loops. The 12 residue test set contains the ten 12-residue loops gathered by Jacobson and colleagues (Jacobson *et al.*, 2004), and the 15 residue test set contains the 30 15-residue loops used in the analysis by Zhao *et al.*, 2011. The full lists of loop systems used in this work are provided in Tables S1-S3.

Table S1. 9-residue Test Set

Loop number	Corresponding PDB numbering
1	3pte-09 (A107-A115)
2	1xyz-09 (A795-A803)
3	1lkk-09 (A193-A201)
4	1mla-09 (A194-A202)
5	2ayh-09 (A169-A177)
6	1arb-09 (A90-A98)
7	1mrp-09 (A284-A292)
8	1cse-09 (E95-E103)
9	1tca-09 (A217-A225)
10	2eng-09 (A172-A180)
11	1xyz-09 (A568-A576)
12	1aba-09 (A69-A77)
13	1nif-09 (A266-A274)
14	1arp-09 (A127-A135)
15	1noa-09 (A99-A107)
16	1lkk-09 (A142-A150)
17	1xif-09 (A59-A67)
18	1rhs-09 (A216-A224)
19	1fus-09 (A91-A99)
20	1php-09 (A91-A99)
21	2cpl-09 (A24-A32)
22	1nls-09 (A131-A139)
23	1xnb-09 (A133-A141)
24	1btl-09 (A102-A110)
25	1mrj-09 (A92-A100)
26	1gpr-09 (A63-A71)
27	1csh-09 (A252-A260)
28	1sgp-09 (E109-E117)
29	3pte-09 (A78-A86)
30	1isu-09 (A30-A38)
31	1noa-09 (A9-A17)
32	2hbg-09 (A18-A26)
33	1nfp-09 (A12-A20)
34	1pgs-09 (A117-A125)
35	1tca-09 (A170-A178)
36	1ptf-09 (A10-A18)
37	1npk-09 (A102-A110)
38	3pte-09 (A215-A223)
39	1ra9-09 (A142-A150)
40	1mrk-09 (A53-A61)
41	1wer-09 (A942-A950)
42	3tgl-09 (A56-A64)
43	2sil-09 (A183-A191)
44	1amp-09 (A57-A65)
45	1aac-09 (A58-A66)
46	1arb-09 (A168-A176)
47	1fus-09 (A31-A39)
48	1byb-09 (A246-A254)
49	1xnb-09 (A116-A124)
50	1ede-09 (A257-A265)
51	1aru-09 (A36-A44)
52	1onc-09 (A70-A78)
53	1noa-09 (A76-A84)

Table S2. 12-residue Test Set

Loop number	Corresponding PDB numbering
54	1arb-12 (A74-A85)
55	1dim-12 (A213-A224)
56	1xyz-12 (A813-A824)
57	1bkf-12 (A9-A20)
58	2ayh-12 (A21-A32)
59	1akz-12 (A181-A192)
60	1luc-12 (A158-A169)
61	153l-12 (A98-A109)
62	1cex-12 (A40-A51)
63	1ixh-12 (A160-A171)

S2.2 Sampling efficiency of basic MoMA-LoopSampler

S2.2.1 Distance to native loop

To test the ability of MoMA-LoopSampler to sample loop configurations close to the native state, we ran a series of tests on each test dataset. The run time for each test was determined

Table S3. 15-residue Test Set

Loop number	Corresponding PDB numbering
64	2v3v-15 (A382-A396)
65	1qqf-15 (A1112-A1126)
66	1h4a-15 (X19-X33)
67	2aeb-15 (B156-B170)
68	3a3p-15 (A286-A300)
69	1wui-15 (L454-L468)
70	1qaz-15 (A298-A312)
71	3css-15 (A95-A109)
72	3a64-15 (A350-A364)
73	2o2k-15 (A1220-A1234)
74	1ju3-15 (A486-A500)
75	2h3l-15 (A1339-A1353)
76	2cjp-15 (A58-A72)
77	1ryo-15 (A172-A186)
78	1ah7-15 (A157-A171)
79	1s95-15 (A477-A491)
80	1ra0-15 (A361-A375)
81	1wb4-15 (A1033-A1047)
82	1y12-15 (A10-A24)
83	3f1l-15 (A99-A113)
84	2pkf-15 (A26-A40)
85	2oit-15 (A290-A304)
86	2dsj-15 (A354-A368)
87	1bhe-15 (A121-A135)
88	3ea1-15 (A136-A150)
89	2b0t-15 (A701-A715)
90	1ra0-15 (A283-A297)
91	1zhx-15 (A392-A406)
92	3bb7-15 (A231-A245)
93	3bf7-15 (A49-A63)

by the loop length being sampled: 2 hours for 9-residue loops, 4 hours for 12-residue loops, and 6 hours for 15-residue loops. Each experiment was repeated 4 times. [1]

Figure S6 gives the distribution of the lowest RMSD to the native loop for both the best and worst of the four executions. In all four tests, at least one conformation within 2 Å of native (using backbone RMSD as a distance metric) was sampled for each of the 9- and 12-residue loops, and between 23 and 26 of the 15-residue loops. Decreasing the threshold to test for a sample within 1 Å of the native state, the results are still very good for 9- and 12-residue loops: with 51 9-residue loops (out of 53) and 9 12-residue loops (out of 10). For the 15-residue loops however, this number drops to 9 loops (over all four tests, this was achieved for 11 loops). The fact that these results vary from one test to another suggests that the sampling time is not sufficient for some 15-residue loops. With such a loop length, the number of possible conformations may be very large when the loop environment is not strongly constrained, and thus, the sampling time must be adapted to the size of the conformational space.

The times observed to sample a loop within 1 or 2 Å of the native are reported in Table S4. Computational requirements increase with loop length. Given that the native loop is one of many valid conformations, the relatively high variance in the time to generate a nearby configuration for long loops is not surprising. This is due to the stochasticity of the method. The variation of RMSD_{min} for 15-residue loops suggests that the provided sampling time is not adequate to perform a sufficient sampling of the loop's conformational space.

[1] Computing times reported throughout the paper correspond to runs on a single core of a 2.30 GHz Intel® Xeon® E5-2695 v3 processor.

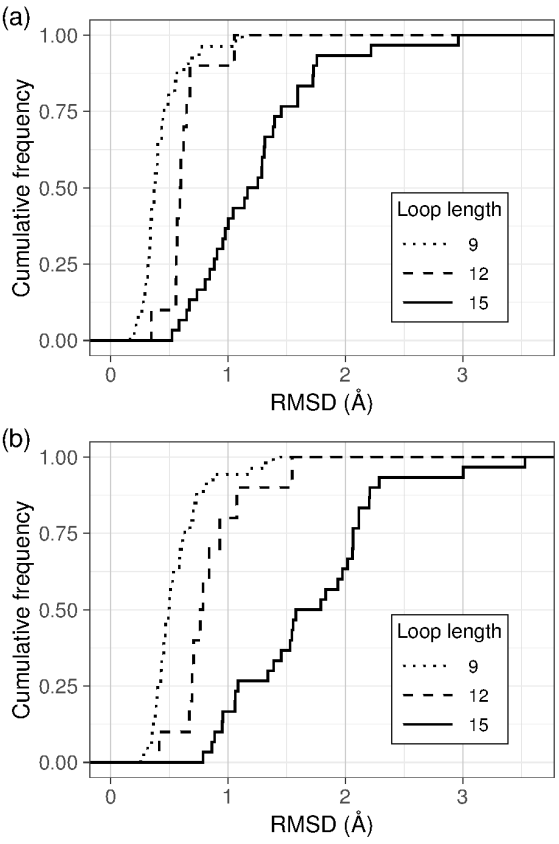


Fig. S6. Cumulative distribution of the lowest backbone RMSD to native among sampled conformations for the 93 loops in our test sets. Since sampling was performed four times, (a) shows the distribution of the best of the four results for each loop, while (b) shows the distribution of the worst of the four results.

These results show that MoMA-LoopSampler can construct loop ensembles for the 9 and 12-residue cases that include the native state with high precision, while 15-residue loops still present a formidable challenge, due to the high dimensionality of the search space possibly coupled with a less constrained environment.

S2.2.2 Comparison of MoMA-LoopSampler with loop prediction methods

We compare the sampling performance of MoMA-LoopSampler to that of DiSGro (Tang *et al.*, 2014) and of the updated version of RCD (Chys and Chacón, 2013; López-Blanco *et al.*, 2016). Note that, based on the results from references (Soto *et al.*, 2008) and (Tang *et al.*, 2014), DiSGro performs better than earlier loop closure or loop prediction methods such as CCD (Canutescu and Dunbrack, 2003), Wriggling (Cahill *et al.*, 2003), PLOP-build (Jacobson *et al.*, 2004), LOOPY_{bb} (Xiang *et al.*, 2002), Random Tweak (Shenkin *et al.*, 1987), or Direct Tweak (Xiang *et al.*, 2002; Xiang, 2006). Therefore, we do not compare directly to these older methods.

Source code for DiSGro was obtained from `http://tanto.bioe.uic.edu/DiSGro/download.html`. The code had to be slightly modified to output exactly the required number of clash-free conformations (instead of the subset of clash-free conformations among a required number of closed ones), and

Table S4. Time needed to generate a conformation within 1 or 2 Å of the native loop. Statistics are calculated on all the runs for which such a distance was reached. MoMA-LoopSampler sampled a loop within 2 Å of native in at least one of the four tests for all 9- and 12-residue loops and 28 15-residue loops. It sampled a loop within 1 Å of native in at least one of the four tests for 51 9-residue loops, 9 12-residue loops and 11 15-residue loops.

Test Set	Time to reach...							
	2 Å to native				1 Å to native			
	Min	Median	Mean	Max	Min	Median	Mean	Max
9 residues	0.3 sec	4.5 sec	21.4 sec	11.8 min	1.0 sec	28.6 sec	4.1 min	1.7 h
12 residues	0.89 sec	23.4 sec	2.5 min	28.9 min	3.26 sec	5.6 min	16.5 min	2.2 h
15 residues	2.84 sec	19.6 min	1 h	5.9 h	4 min	1 h	1.6 h	5.1 h

conformations were generated without side-chains. Binaries for RCD version 1.40 were downloaded from `http://chaconlab.org/modeling/rcd/rcd-download`. MoMA-LoopSampler uses stricter constraints than DiSGro and RCD, in particular for steric clash detection. Therefore, in order to more adequately compare running times, we also tested a variant of MoMA-LoopSampler (Soft MoMA-LoopSampler) that uses collision constraints comparable to that of DiSGro and RCD. This variant uses a van der Waals scaling factor of 0.6 (instead of 0.7 for the other tests), does not use enlarged C_β atoms, and uses a lower max_{IK}. These changes are expected to lower the quality of the ensemble and its exhaustiveness, but also to considerably decrease sampling time, allowing a more straightforward time comparison with DiSGro and RCD. The computational time and ability to generate near-native loops are compared, using the same computational resources for all four methods.

Difference between the sampling methods

Different sets of constraints are enforced by the three sampling methods. Concerning collisions, DiSGro employ an energy function that makes steric clashes unlikely. The maximum allowed ratio between non-bonded atom pair distances and the sum of their van der Waals radii (called van der Waals scaling factor) was set to 0.6 in DiSGro (value found in the source code) and to 0.5 for intra-loop backbone collisions in RCD. Collisions with the rest of the protein are handled differently in RCD: this method uses a grid and considers that there is a collision if an atom of the loop is placed in a non-empty cell. This collision detection method is much faster but also less accurate than considering the actual distance between atoms. In MoMA-LoopSampler, steric clash avoidance is a crucial component: backbone atoms and the C_β atoms (with enlarged volumes to account for side-chain placement) of the sampled loops are placed without major steric clash among themselves or with the rest of the protein. While the van der Waals scaling factor was set to 0.6 in the more collision-tolerant version of MoMA-LoopSampler (Soft MoMA-LoopSampler), we set this cutoff at 0.7 to test the basic and H_{RL} MoMA-LoopSampler.

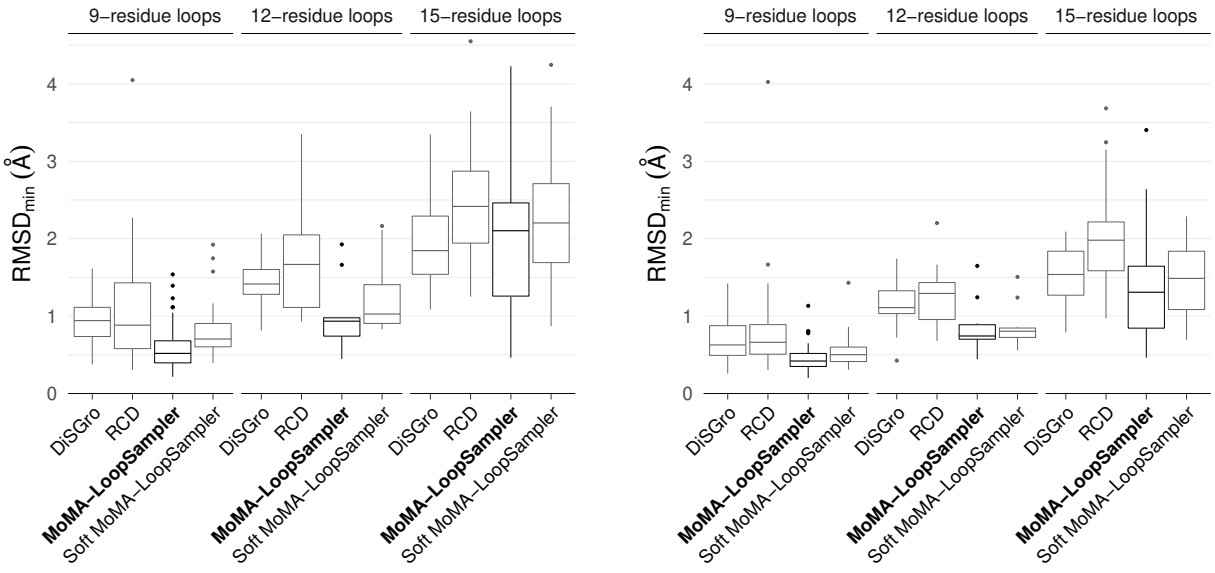
Structural knowledge is included in all three methods. Although RCD can be considered an *ab initio* method, it samples dihedral angles following neighbor-dependent Ramachandran probability distributions. DiSGro includes a stronger knowledge-based component, with a more complex dihedral angle sampling that follows distributions extracted from a structural database. Finally, MoMA-LoopSampler is strongly dependent on structural knowledge, since it directly uses fragments from experimentally-solved protein structures.

Results

Among 5000 sampled conformations, RMSD_{min} obtained by MoMA-LoopSampler are much lower than that obtained by the other methods for 9- and 12-residue loops (Figure S7(a)). Soft MoMA-LoopSampler also obtains RMSD_{min} lower than DiSGro and RCD, but higher than the basic version of MoMA-LoopSampler. For the 15-residue loops, RMSD_{min} obtained on 5,000 sampled conformations are comparable for the four methods. However, looking at 100,000 sampled conformations for these longer loops, MoMA-LoopSampler obtains a much lower RMSD_{min} (Figure S7(b)). Note that the RMSD_{min}s obtained for DiSGro on the 15-residue loop test set are lower than the ones the authors report for 100,000 sampled conformations on the same test set (Tang *et al.*, 2014).

Generating 100,000 conformations instead of 5,000 lowered the RMSD_{min} for all the methods (Figure S9). However, this decrease varies from one method to another. It is very limited for MoMA-LoopSampler on the 9- and 12-residue loops, while being considerable for other methods. Note that despite this, the RMSD_{min} obtained by MoMA-LoopSampler on these loops is still much lower compared to the other tested methods. The 15-residue loops show the opposite trend: the decrease in RMSD_{min} is much larger for MoMA-LoopSampler and its "soft" version than for the other methods on 15-residue loops, resulting in a lower RMSD_{min} for MoMA-LoopSampler compared to RCD and DiSGro.

Concerning running times, MoMA-LoopSampler and its soft version are much less sensitive to the length of the loop than the other two methods. The soft version of MoMA-LoopSampler is slightly slower than other methods on 9-residue loops but faster than DiSGro for 12- and 15-residue loops. We note that the running times obtained for DiSGro are higher that those reported in Tang *et al.*, 2014. In some tests, DiSGro got blocked during the sampling process, in which case we started the run again. These failed sampling attempts are not counted in the running times we report. The basic version of MoMA-LoopSampler is unsurprisingly slower than the other methods, due to the stronger constraints it enforces. Overall, results show that the "soft" version of MoMA-LoopSampler has running times comparable to that of other methods and provides slightly lower RMSD_{min}. The basic version of MoMA-LoopSampler on the other hand, trades computational efficiency off for better filtering of sampled ensembles.



(a) 5,000 sampled conformations.

(b) 100,000 sampled conformations.

Fig. S7. Minimum distance to native (RMSD$_{min}$) obtained among sampled conformations, without side-chains.

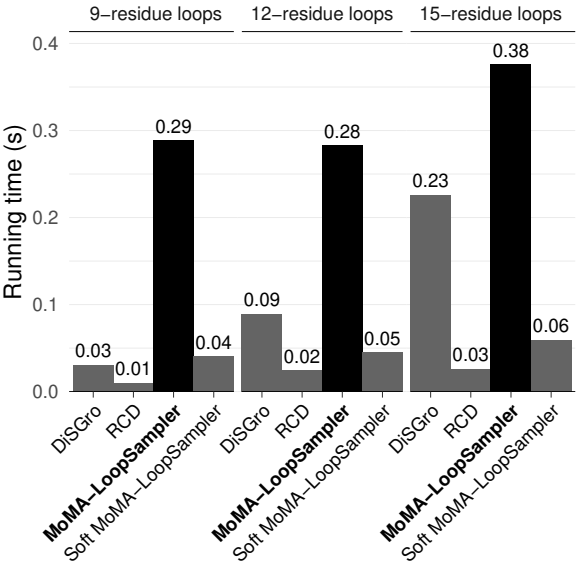


Fig. S8. Median time per sampled conformation (estimated on 5000 sampled conformations, without side-chain placement). Computations were performed using a single core of a 2.30 GHz Intel® Xeon® E5-2695 v3 processor.

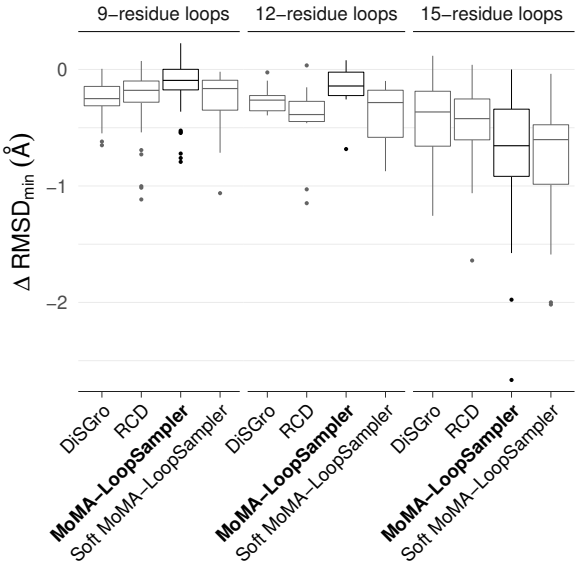


Fig. S9. Difference in RMSD$_{min}$ obtained when generating a larger number of conformations (100,000 conformations instead of 5,000).

Discussion

The difference in the collision constraints enforced by the different methods can help to explain two major observations in the results: (1) MoMA-LoopSampler (especially the basic version) is less sensitive to loop length than RCD and DiSGro, from a running time point of view; (2) while MoMA-LoopSampler obtains a much lower RMSD$_{min}$ than other methods for 9- and 12-residue loops on ensembles of 5,000 conformations, a higher number of sampled conformations is necessary for 15-residue loops to observe a difference.

To explain the first point, we hypothesize that MoMA-LoopSampler is more sensitive to the environment of the loop, and whether it is constrained or not, than to the length of the loop. First, an essential component of the method consists in checking that the distance between the two working loop ends can be covered by the tripeptides left to place. Making this verification after a tripeptide is added facilitates the closing of long loops. Second, shorter loops generally have a more constrained environment than longer loops. As previously mentioned, MoMA-LoopSampler is more intolerant to collisions compared to other methods. Therefore, generating a conformation for a loop in a more constrained environment is a problem with a difficulty comparable to that of sampling a longer unconstrained loop for MoMA-LoopSampler, which

explains why median running times per conformation vary little with loop length.

To explain the second point, a similar reasoning can be conducted. MoMA-LoopSampler only samples the accessible conformational space by carefully avoiding collisions in generated conformations. Therefore, it finds the native conformations using fewer samples than other methods. With more constrained environments, 9- and 12- residue loops have a resulting conformational space that is particularly reduced, which is why MoMA-LoopSampler finds the native conformation very early into the search. Longer loops are usually more flexible, and a larger portion of the conformational space is allowed. Therefore, methods that are overall more tolerant to collisions sample fewer bad-quality conformations in proportion. For these loops, more conformations need to be sampled to achieve a better coverage of the conformational space. The benefit in RMSD_{min} for MoMA-LoopSampler is thus logically observable when sampling a higher number of conformations (100,000) for 15-residue loops (Figure S7(b)).

In simpler terms, this means that MoMA-LoopSampler better explores the conformational space, performing an exhaustive exploration using fewer samples than other methods. The difference in RMSD_{min} observed after generating larger ensembles further supports this idea. Indeed, 5,000 samples from MoMA-LoopSampler are enough to explore the conformational space of 9- and 12-residue loops, explaining why the RMSD_{min} barely decreases when generating a much larger number of conformations. For other methods, the RMSD_{min} considerably decreases upon generating more conformations, showing that these methods keep discovering relevant conformations among the extra conformations. Conversely, for 15-residue loops, the difference in RMSD_{min} obtained upon generating 100,000 conformations instead of 5,000 is considerable for all four methods, with MoMA-LoopSampler and Soft MoMA-LoopSampler showing a much larger decrease than RCD and DiSGro. This confirms (1) that 5,000 conformations are not enough to cover the much larger conformational space of these longer loops, and (2) that MoMA-LoopSampler performs a more efficient exploration, discovering relevant conformations using fewer samples than RCD and DiSGro.

Obtaining an ensemble of conformations of good quality is essential considering the costly downstream processing steps of applications involving loop sampling, in particular in the context of stable states prediction. These steps include side-chain addition, relaxation, scoring, clustering or filtering, and can be extremely time-consuming. In that regard, generating fewer conformations, but which are more representative of the ensemble overall, is perfectly satisfactory. This suggests that MoMA-LoopSampler is a good candidate for the sampling stage of many structural bioinformatics applications, including stable states prediction, since it obtains the same RMSD_{min} as other methods (or a lower one) without needing to sample as many conformations as these methods do.

S2.3 Application to a multi-state loop: complements

S2.3.1 Post-processing of streptavidin loop conformations

Several experimentally-determined structures were gathered for streptavidin and sampling for the flexible loop was performed from each of them. For all the conformations obtained after three rounds of brute force search, side-chain placement was attempted with an in-house method using continuous rotamers from BASILISK (Harder *et al.*, 2010). Only conformations for which side-chains could be placed without steric clash were retained and relaxed using energy minimization protocols from Amber 16 (Case *et al.*, 2005, 2016). This represents 5338 conformations from scaffold 2F01 chain A, 3825 from 2F01 chain B, 4042 from 3RY1 chain A, 1304 from 3RY1 chain B, 702 from 3RY1 chain C, 2820 from 3RY1 chain D, 5320 from 3RY2 chain A and 3611 from 3RY2 chain B. No constraints were applied for the relaxation, so that both backbone and side-chain movements were allowed. Bond length, bond angles and dihedral angles were all free to move. The relaxation took 133 s per loop on average. The first 250 cycles used steepest descent minimization, while the remaining cycles applied conjugate gradient. The maximum number of cycles was set to 500 and the minimization was considered to have converged when the root-mean-square of the cartesian elements of the gradient was lower than 0.1 kcal/(mol·Å). Energy was calculated using the ff14SBonlysc force field and a simple Generalized Born implicit solvent model ($igb = 1$ and the *mbondi* radii sets, as recommended in the Amber manual).

S2.3.2 Detailed analysis of the energy landscape of streptavidin's flexible loop

The projection plots show major differences depending on the starting structure. Sometimes, both the "open" and "closed" conformations appear to be in low-energy basins (2F01-A, 2F01-B, 3RY2-A, 3RY1-B and 3RY1-C). Other projection plots show only one low-energy basin, around the crystallographic conformations (3RY2-B, 3RY1-A and 3RY1-D). The energies were calculated without a ligand, showing the strong influence of the local structure around the loop. In the case of 2F01-A, 2F01-B, and 3RY2-A, the environment surrounding the loop allows it to adopt both conformations, but the presence of the ligand probably stabilizes the loop in one of the two basins. Energy barriers of different heights separate the "closed" and "open" basins. In 3RY2-B, it seems that only the "closed" conformation is stable, suggesting that the environment of the loop also changes (due to crystal packing or ligand binding), and stabilizes this conformation. A profile similar to that of 3RY2-B is found for 3RY1-A, although this subunit is unbound in the crystallographic structure. This is an indication of the conformational changes that occur around the loop in the crystal structure. For 3RY1-D, the "open" conformation is in a large low energy basin. A few low energy conformations are found in the "closed" loop region but a large, high energy barrier separates the two regions. In the case of 3RY1-B and 3RY1-C, the conformational space appears tighter. The loop environment is probably more constrained sterically. Nevertheless, both energy basins are found, with a lower energy barrier separating them, and a much lower energy minimum for the "open" basin, explaining the "open" conformation adopted by these subunits in the crystal.

S2.3.3 Landscapes obtained using other sampling methods

The protocol for modeling and visualizing the energy landscape of streptavidin (Section 3.2.2) was repeated using DiSGro

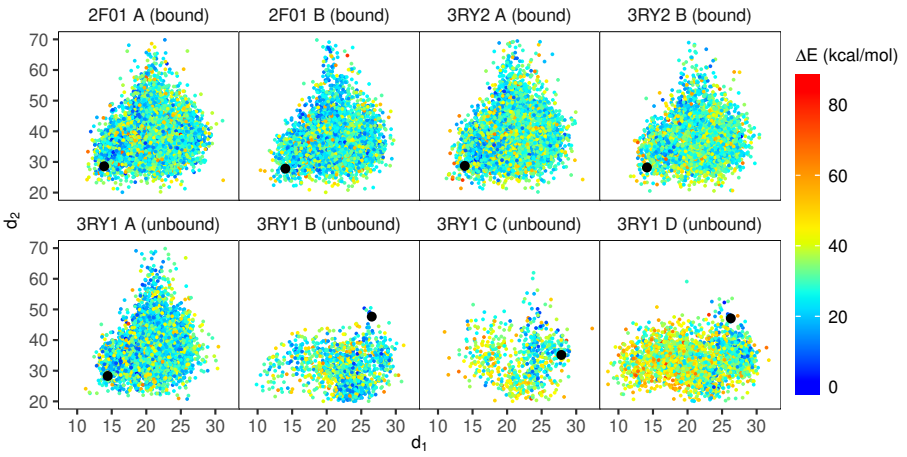


Fig. S10. 2D projections of conformations sampled using DiSGRO for a loop in the streptavidin protein, from eight starting X-ray crystallography protein structures. The first dimension, d_1 (x-axis), is the distance (Å) between an atom located in the middle of the loop and a fixed atom in the protein. The second dimension, d_2 (y-axis), is the angle (degrees) formed by three atoms: an atom at approximately one quarter, one half, and three quarters of the way down of the loop. The conformations from the crystallographic structures are shown in black. For each system, the loop with the lowest energy was identified and each conformation was then colored according to the difference between its energy and this lowest energy.

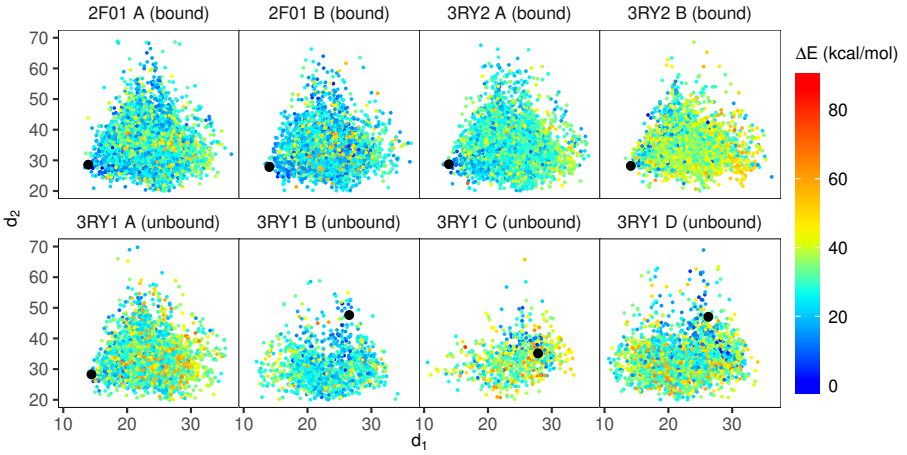


Fig. S11. 2D projections of conformations sampled using RCD+ for a loop in the streptavidin protein, from eight starting X-ray crystallography protein structures. The first dimension, d_1 (x-axis), is the distance (Å) between an atom located in the middle of the loop and a fixed atom in the protein. The second dimension, d_2 (y-axis), is the angle (degrees) formed by three atoms: an atom at approximately one quarter, one half, and three quarters of the way down of the loop. The conformations from the crystallographic structures are shown in black. For each system, the loop with the lowest energy was identified and each conformation was then colored according to the difference between its energy and this lowest energy.

and RCD+ (López-Blanco *et al.*, 2016) as sampling methods. RCD+ is a web server that performs loop modeling using RCD as a first sampling step and then performs side-chain placement and refinement. For each scaffold, DiSGRO and RCD+ were employed to sample the same number of conformations as sampled using MoMA-LoopSampler in brute force mode. Side-chain placement was activated for DiSGRO. The sampled conformations were relaxed and projected in 2D. The resulting landscapes are shown in Figures S10 and S11.

The landscapes are very different from those obtained using MoMA-LoopSampler. Although some common features can be observed (such as the basin around the "open" conformations from scaffolds 3RY1 B, C and D), the landscapes are much rougher and harder to interpret. They are also more spread than landscapes obtained when sampling with MoMA-LoopSampler. The fact that these methods are more collision-tolerant may explain these observations. Indeed, many statistically unlikely conformations are generated, perturbing the analysis of the landscape. By creating a better filtered ensemble, MoMA-LoopSampler clarifies the analysis of energy landscapes for this loop.

S2.4 Effects of H_{RL} on the generated ensembles

S2.4.1 Influence of learning rate over the number of sampled conformations

Figure S12 shows that the higher the learning rate, the larger the number of conformations sampled. This is true for many

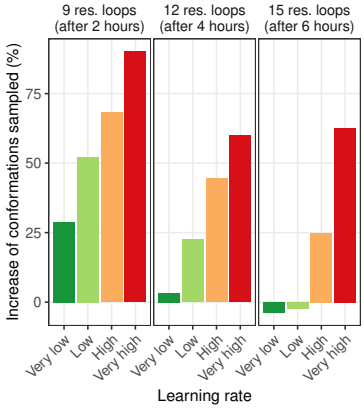


Fig. S12. Median percentage of increase in number of conformations sampled in H_{RL} modes compared to the basic mode across loop systems.

systems such as loops 14 and 68 (Figure S13), which show very different sampling speed depending on the learning rate. However, the effect of H_{RL} depends on both the length of the loop and the loop/protein system itself. While most 9- and 12-residue loops (and a few 15-residue loops) exhibit this expected behavior, for other loops (and for many 15-residue loops), only runs performed with *high* and *very high* learning rates are capable of generating a larger number of conformations (e.g. loop 61). As mentioned in the main paper, activating H_{RL} with a *very low* or *low* learning rate can even reduce

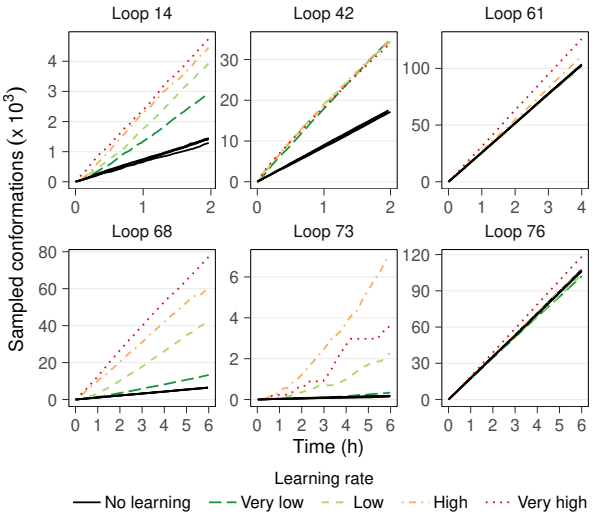


Fig. S13. Number of conformations sampled as a function of time for different learning rates for a few representative loops.

the number of sampled conformations, compared to the basic mode. This is mainly observed for 15-residue loops, such as loop 76. Loop 42 illustrates yet another case: all learning rates generate loops at comparable speed, but still much faster than MoMA-LoopSampler in basic mode does.

Loops 68 and 73 also illustrate an interesting phenomenon. The curves for *high* and *very high* learning rates show some plateaus spanning 10 minutes or more, which are due to the overhead of maintaining a very large learning tree, whose dimensionality grows exponentially in the length of the loop.

S2.4.2 Time needed to sample near-native conformations

The cumulative distribution of the time necessary to generate the first conformation within 1 Å RMSD to native, with or without learning, is shown Figure S14. Although these values are generally of the same order of magnitude for runs with or without learning, some observations can still be made. Looking at results loop by loop, we observe that for 9-residue loops, using H_{RL} may considerably delay the sampling of a conformation that is close to native. A possible reason is that introducing H_{RL} modifies the probability for selecting tripeptides. While in the basic mode, MoMA-LoopSampler picks tripeptide states at each step with a uniform distribution, the H_{RL} mode offers MoMA-LoopSampler the possibility to adjust the sampling of states so that a suitable distribution is obtained. Considering the tree used by H_{RL} to organize the tripeptides, MoMA-LoopSampler in the basic mode chooses each cell with a distribution that is directly proportional to the number of tripeptides it contains. Conversely, in the learning mode, MoMA-LoopSampler samples each cell according to their score. The score is currently set so as to sample effectively as many diverse conformations as possible, but other strategies may be contemplated in order to obtain a loop ensemble that follows the density of the tripeptide database for each tripeptide position. Such an ensemble could provide a more statistically accurate representation of the loop conformational space, which would be interesting to analyze entropic effects.

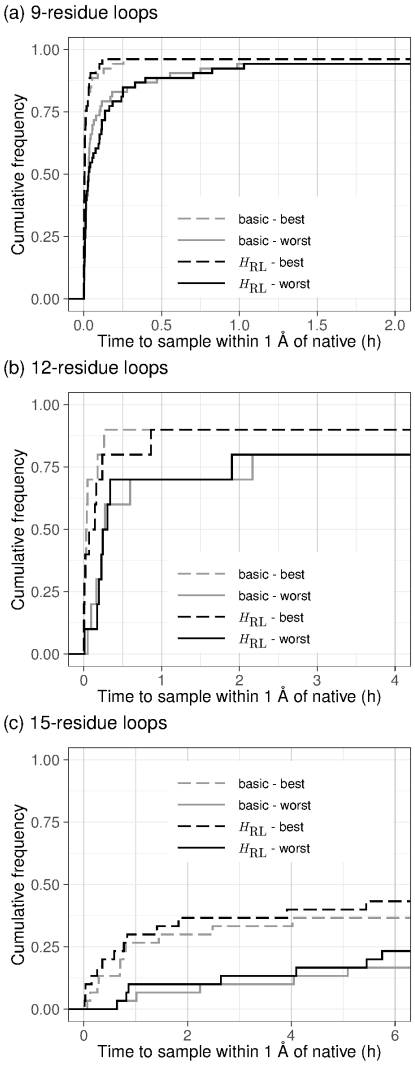


Fig. S14. Cumulative distribution of the time needed to sampled a conformation within 1 Å of the native for the 93 loops in our test sets. Four distributions were calculated: one using the shortest time of the four tests in basic mode for each loop, one using the longest time of the four tests in basic mode for each loop, one using the shortest time of the four tests in H_{RL} mode for each loop, and finally one using the longest time of the four tests in H_{RL} mode for each loop.

S2.4.3 Effect of H_{RL} on RMSD$_{min}$

Main results and discussions regarding the effect of H_{RL} on the value of RMSD$_{min}$ for the loop test-sets are presented in the manuscript. As a complement to these results, Table S5 gives the mean and median values obtained for RMSD$_{min}$.

S2.4.4 Evolution of sampling in time

We also show the evolution of the distribution of sampled loops with H_{RL} activated in Figure S15. These heatmaps, shown in the same two-dimensional projection, showcase the density of the sampled loops in the first and the last ten minutes of the exploration. Two effects are observed when comparing the beginning and the end of the sampling process:

(1) The first effect of learning is that the number of sampled conformations increases, and the coverage of the conformational space improves. In other words, the projection of sampled loop conformations in 2D appears to be more homogeneous and continuous. The effect is similar to that

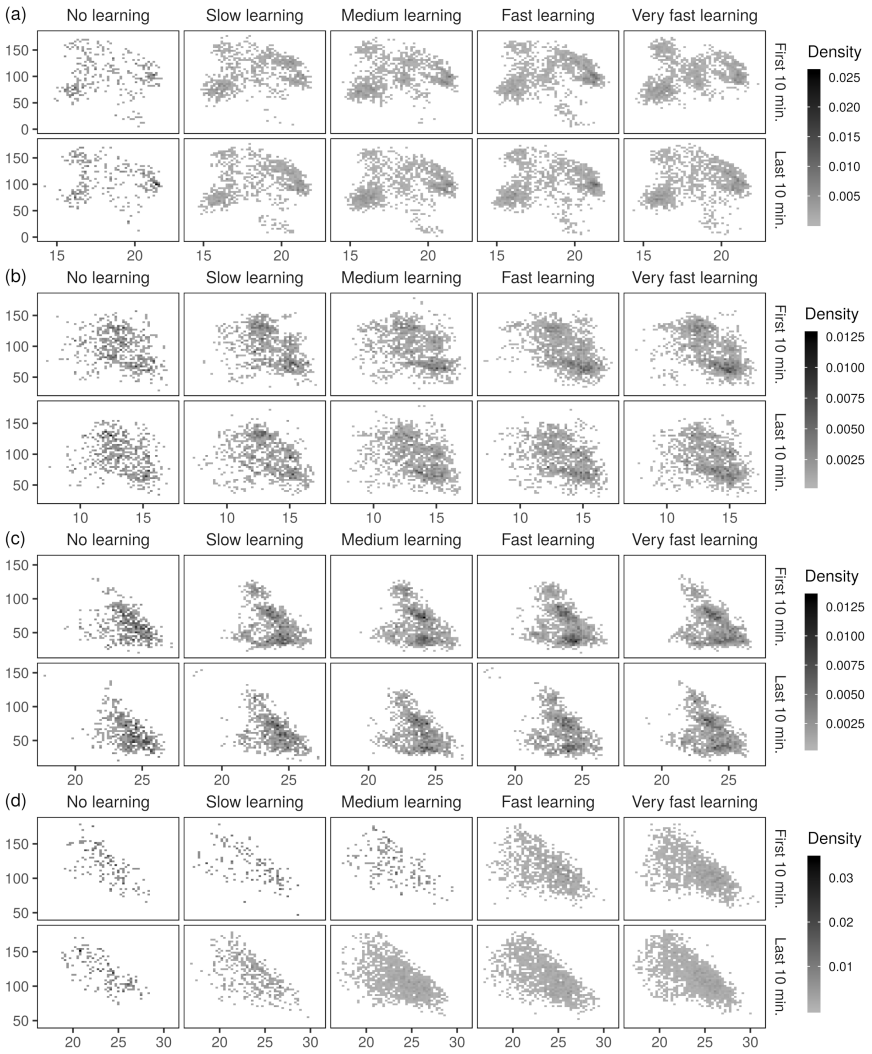


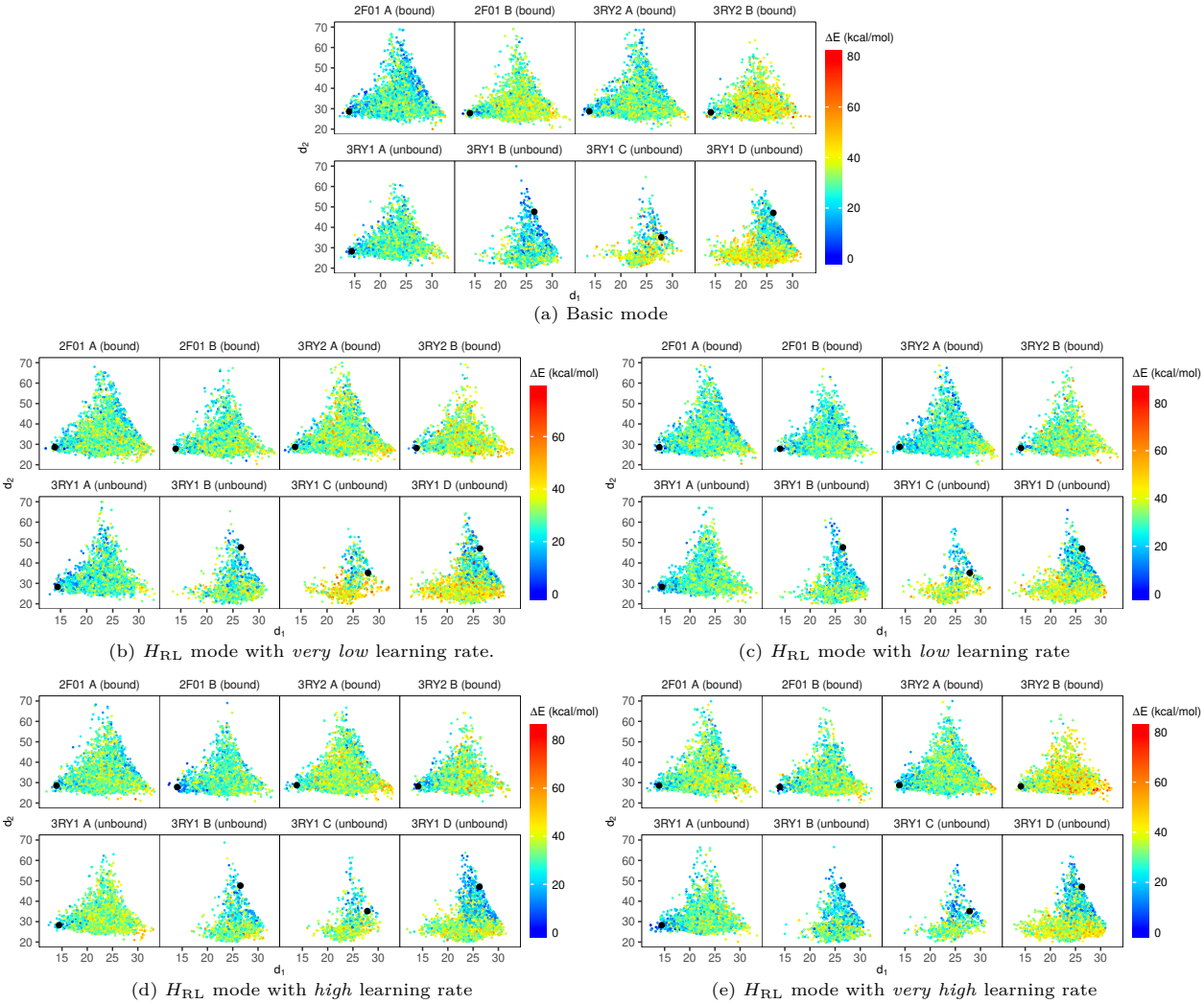
Fig. S15. Evolution of the distribution of sampled loops due to H_{RL}. The plots show heatmaps of two-dimensional projections of the sampled loops for different learning rates, during the first and the last ten minutes of sampling. The x-axis gives the first projection descriptor d_1 (in Å), while the y-axis gives the second projection descriptor, d_2 (in degrees). (a) Loop 21, (b) Loop 26, (c) Loop 40, (d) Loop 68.

Table S5. RMSD$_{min}$ obtained for different learning rates.

Length		Basic mode	H_{RL} mode learning rate			
			V. low	Low	High	V. high
9 res.	Mean (Å)	0.49	0.48	0.47	0.46	0.46
	Median (Å)	0.43	0.41	0.41	0.39	0.41
	SD (Å)	*0.21*	*0.21*	*0.21*	*0.20*	*0.19*
12 res.	Mean (Å)	0.73	0.67	0.73	0.67	0.75
	Median (Å)	0.67	0.60	0.64	0.55	0.69
	SD (Å)	*0.26*	*0.20*	*0.30*	*0.32*	*0.26*
15 res.	Mean (Å)	1.50	1.52	1.48	1.48	1.50
	Median (Å)	1.45	1.45	1.43	1.42	1.44
	SD (Å)	*0.62*	*0.68*	*0.70*	*0.67*	*0.73*

observed for the different learning rates. Indeed, the algorithm progressively stops exploring the regions of space where it does not find any solution. The probability to sample a tripeptide in a cell from which all attempts have failed so far decreases with running time. Consequently, the success rate becomes higher since MoMA-LoopSampler focuses on the vicinity of regions that are successful. This is very clear in Figure S15(d), for *very low* and *low* learning rates. It can also be observed for other systems, although to a lesser degree. The ability of H_{RL} to quickly identify areas where no solution exists depends on the positions of solutions in the conformational space and how they cluster, on the projection chosen to organize the tripeptide states, and on the learning rate.

(2) The second effect is the sudden discovery of whole regions of the conformational space. As previously mentioned, the probability to explore a region invariably found unsuccessful so far decreases based on the the number of attempts and the learning rate. If the number of attempts is too low (the learning process is too greedy), MoMA-LoopSampler can fail to explore some regions in which a few successful conformations could have been found. However, these regions may suddenly get "unlocked" after a closed loop is finally sampled. One successful conformation is necessary and sufficient to set the score of the cell leading to sampling in that region back to its maximum. This is the case for the region



(a) Basic mode

(b) H_{RL} mode with *very low* learning rate.

(c) H_{RL} mode with *low* learning rate

(d) H_{RL} mode with *high* learning rate

(e) H_{RL} mode with *very high* learning rate

Fig. S16. 2D projections of conformations sampled using MoMA-LoopSampler at different learning rates for a loop in the streptavidin protein, from eight starting X-ray protein structures. The first dimension, d_1 (x-axis), is the distance (Å) between an atom located in the middle of the loop and a fixed atom in the protein. The second dimension, d_2 (y-axis), is the angle (degrees) formed by three atoms: an atom at approximately one quarter, one half, and three quarters of the way down of the loop. The conformations from the crystallographic structures are shown in black. For each system, the loop with the lowest energy was identified and each conformation was then colored according to the difference between its energy and this lowest energy.

of the conformational space that is projected in the bottom right-hand corner of the heatmaps shown in Figure S15(a) at a *very high* learning rate. This phenomenon is also observed for other systems in Figure S15, albeit on smaller regions of the projection plot. This is a warning that learning is already too greedy. In theory, learning should only stop exploring regions when enough exploration has been made and the probability to find a successful loop conformation in that region is negligible. In practice, this can never be determined with certainty, as long as the full region has not been explored. For H_{RL} to have the desired behavior, a trade-off has to be found in order to enable faster exploration while not risking to lose large and/or relevant areas of the conformational space.

These results also show that the distribution of sampled conformations may be different depending on the learning rate and the running time: beyond the observation that the conformational space obtained is more continuous, the density observed in the heatmaps evolves between the different conditions. The samples appear to be more uniformly distributed when H_{RL} is involved. Another interesting observation is that, for all these four systems, some consequences of learning become visible very early in the sampling process, within the first 10 minutes.

S2.5 Energy landscape of a multi-state loop at different learning rates

The protocol for modeling and visualizing the energy landscape of streptavidin (Section 3.2.2) was repeated using different learning rates. For each scaffold, MoMA-LoopSampler in basic mode and in four different H_{RL} modes was employed to sample the same number of conformations as sampled using brute force. The sampled conformations were relaxed and projected in 2D. The resulting landscapes are visible in Figure S16.

The landscapes look very similar for all levels of learning, showing that H_{RL} parameterization overall preserves sampling diversity for this loop system. Only slight differences are observed when applying H_{RL} with a *very high* learning rate. In the landscape built from scaffold 2F01 A, the basin around the "open" conformation is not as clearly apparent as for less greedy H_{RL} levels.

References

Cahill, S., Cahill, M., and Cahill, K. (2003). On the kinematics of protein folding. *J. Comput. Chem.*, **24**(11), 1364–1370.

Canutescu, A. A. and Dunbrack, R. L. (2003). Cyclic coordinate descent: A robotics algorithm for protein loop closure. *Protein Sci.*, **12**(5), 963–972.

Case, D., Betz, R., Cerutti, D., T.E. Cheatham I., Darden, T., Duke, R., Giese, T., Gohlke, H., Goetz, A., Homeyer, N., Izadi, S., Janowski, P., Kaus, J., A. Kovalenko, T. L., LeGrand, S., Li, P., C.Lin, Luchko, T., Luo, R., Madej, B., Mermelstein, D., Merz, K., Monard, G., Nguyen, H., Nguyen, H., I.Omelyan, Onufriev, A., Roe, D., Roitberg, A., Sagui, C., Simmerling, C., Botello-Smith, W., Swails, J., Walker, R., Wang, J., Wolf, R., Wu, X., Xiao, L., and Kollman, P. (2016). Amber 2016, University of California, San Francisco.

Case, D. A., Cheatham, T. E., Darden, T., Gohlke, H., Luo, R., Merz, K. M., Onufriev, A., Simmerling, C., Wang, B., and Woods, R. J. (2005). The Amber biomolecular simulation programs. *J. Comput. Chem.*, **26**(16), 1668–1688.

Chys, P. and Chacón, P. (2013). Random Coordinate Descent with Spinor-matrices and Geometric Filters for Efficient Loop Closure. *J. Chem. Theory Comput.*, **9**(3), 1821–1829.

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

Coutsias, E., Seok, C., Jacobson, M., and Dill, K. (2004). A kinematic view of loop closure. *J. Comput. Chem.*, **25**(4), 510–528.

DePristo, M. A., de Bakker, P. I. W., Lovell, S. C., and Blundell, T. L. (2003). Ab initio construction of polypeptide fragments: Efficient generation of accurate, representative ensembles. *Proteins*, **51**(1), 41–55.

Dinner, A. R. (2000). Local deformations of polymers with nonplanar rigid main-chain internal coordinates. *J. Comput. Chem.*, **21**(13), 1132–1144.

Fox, N. K., Brenner, S. E., and Chandonia, J.-M. (2014). SCOPe: Structural Classification of ProteinsâŁ"extended, integrating SCOP and ASTRAL data and classification of new structures. *Nucleic Acids Res.*, **42**(D1), D304–D309.

Harder, T., Boomsma, W., Paluszewski, M., Frellsen, J., Johansson, K. E., and Hamelryck, T. (2010). Beyond rotamers: a generative, probabilistic model of side chains in proteins. *BMC Bioinf.*, **11**(1), 306.

Jacobson, M. P., Pincus, D. L., Rapp, C. S., Day, T. J. F., Honig, B., Shaw, D. E., and Friesner, R. A. (2004). A hierarchical approach to all-atom protein loop prediction. *Proteins*, **55**(2), 351–367.

Kabsch, W. and Sander, C. (1983). Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, **22**(12), 2577–2637.

López-Blanco, J. R., Canosa-Valls, A. J., Li, Y., and Chacón, P. (2016). RCD+: Fast loop modeling server. *Nucleic Acids Res.*, **44**(W1), W395–W400.

Manocha, D. and Canny, J. F. (1994). Efficient inverse kinematics for general 6r manipulators. *IEEE Trans. Robot. Autom.*, **10**(5), 648–657.

Shenkin, P. S., Yarmush, D. L., Fine, R. M., Wang, H. J., and Levinthal, C. (1987). Predicting antibody hypervariable loop conformation. I. Ensembles of random conformations for ringlike structures. *Biopolymers*, **26**(12), 2053–2085.

Soto, C. S., Fasnacht, M., Zhu, J., Forrest, L., and Honig, B. (2008). Loop modeling: Sampling, filtering, and scoring. *Proteins*, **70**(3), 834–843.

Tang, K., Zhang, J., and Liang, J. (2014). Fast Protein Loop Sampling and Structure Prediction Using Distance-Guided Sequential Chain-Growth Monte Carlo Method. *PLoS Comput. Biol.*, **10**(4), e1003539.

Xiang, Z. (2006). Advances in Homology Protein Structure Modeling. *Curr. Protein Pept. Sci.*, **7**(3), 217–227.

Xiang, Z., Soto, C. S., and Honig, B. (2002). Evaluating conformational free energies: the colony energy and its application to the problem of loop prediction. *Proc. Natl. Acad. Sci. U. S. A.*, **99**(11), 7432–7437.

Zhao, S., Zhu, K., Li, J., and Friesner, R. A. (2011). Progress in Super Long Loop Prediction. *Proteins*, **79**(10), 2920–2935.

Appendix: heatmaps comparing tripeptide projections

Following are the figures mentioned in Section S1.7 comparing the different tripeptide state projections.

Fig. S17. Comparison of projections. Several different projections were tested for H_{RL}. The following heatmaps show the ability of each projection to distribute tripeptides and to regroup states that succeed in creating a closed loop together. The results are presented for 6 different systems. For each system, the heatmaps of the learning trees corresponding to the different loop plans are shown. A loop plan is designated by the position of the last tripeptide used to close the loop (IK position). The first of the two heatmaps for a given learning tree gives the distribution of tripeptides in the top level cells of the root tree. The second shows the success probability of the leaf, meaning the number of successful loop conformations that start with a state from this leaf divided by the theoretical number of tripeptide combinations starting with a state from this leaf.

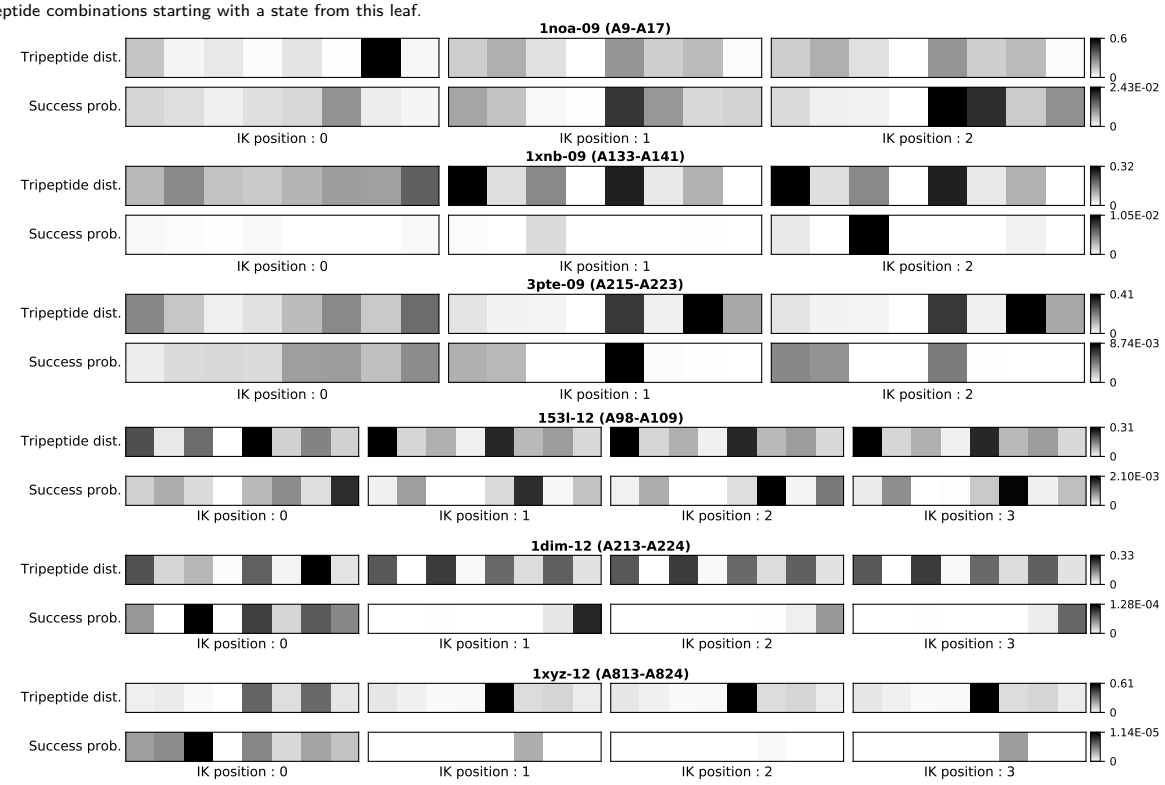


Fig. S17((a)): Position

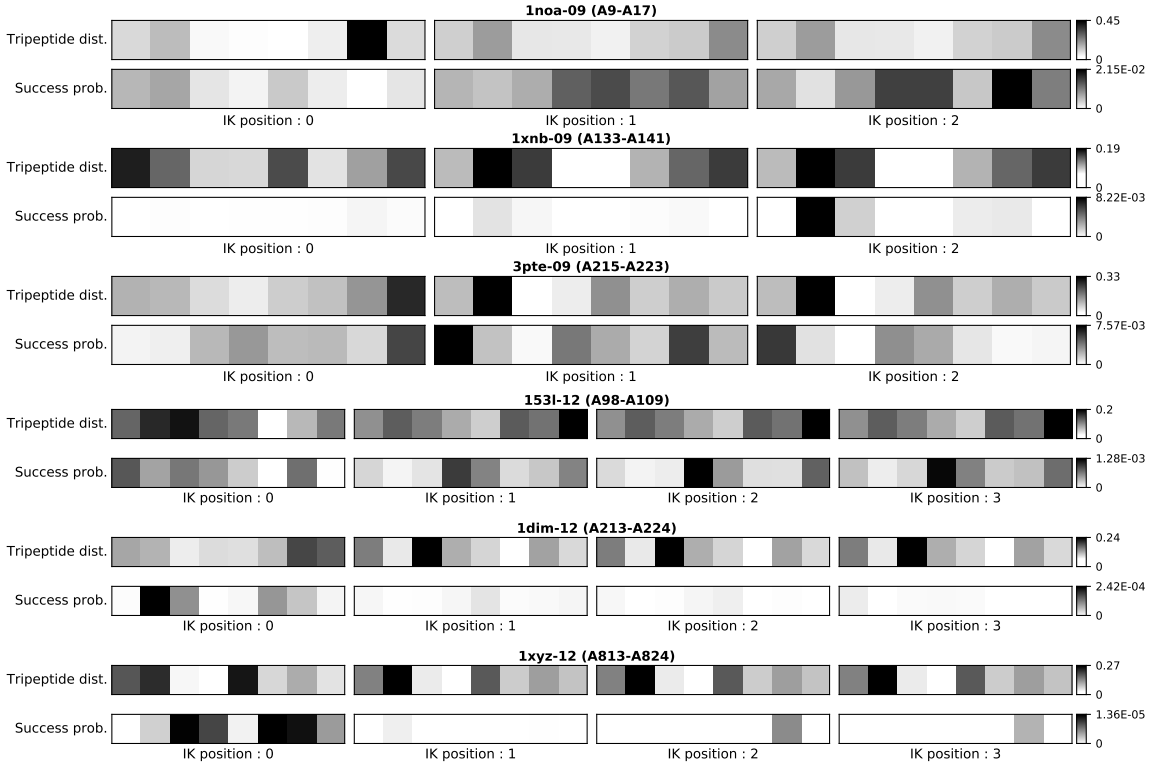


Fig. S17((b)): Euler angles

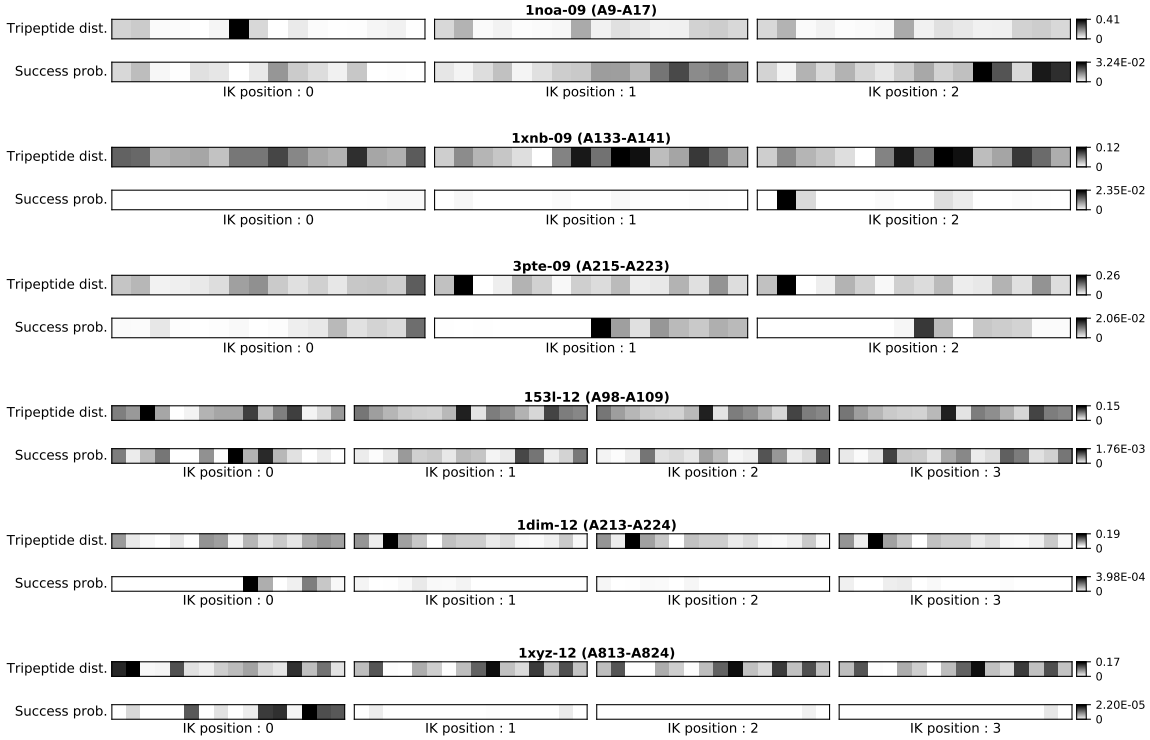


Fig. S17((c)): Euler angles and length

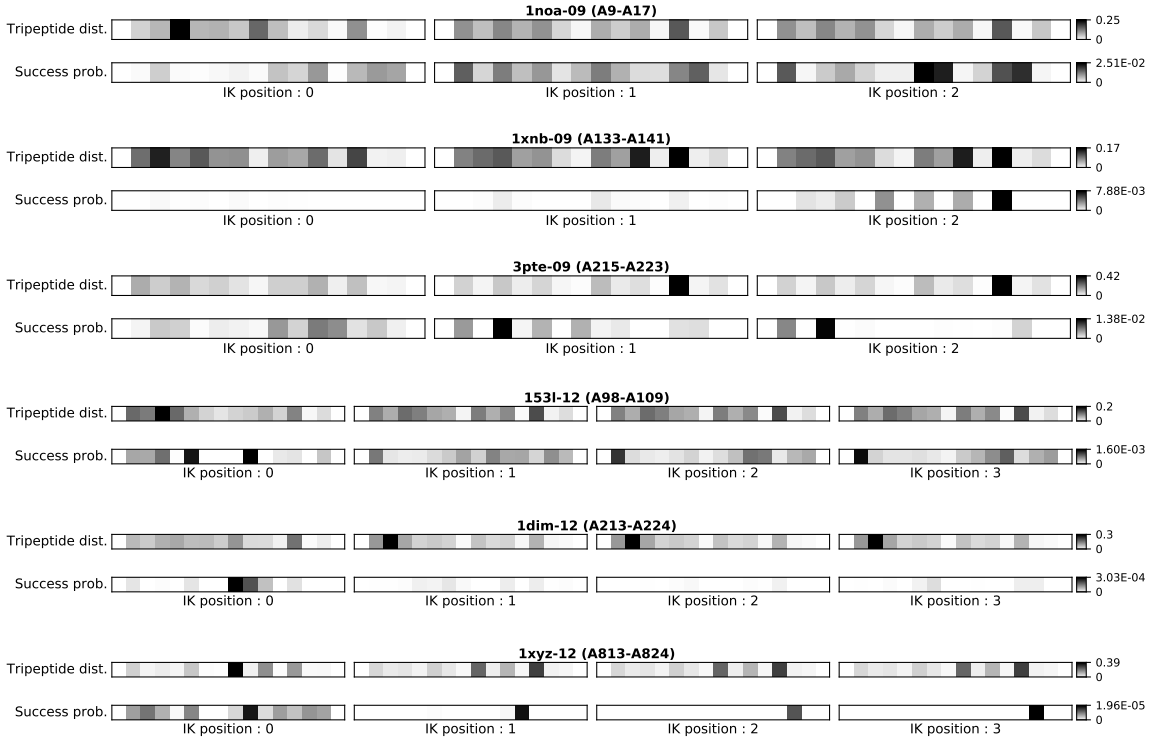


Fig. S17((d)): Quaternion

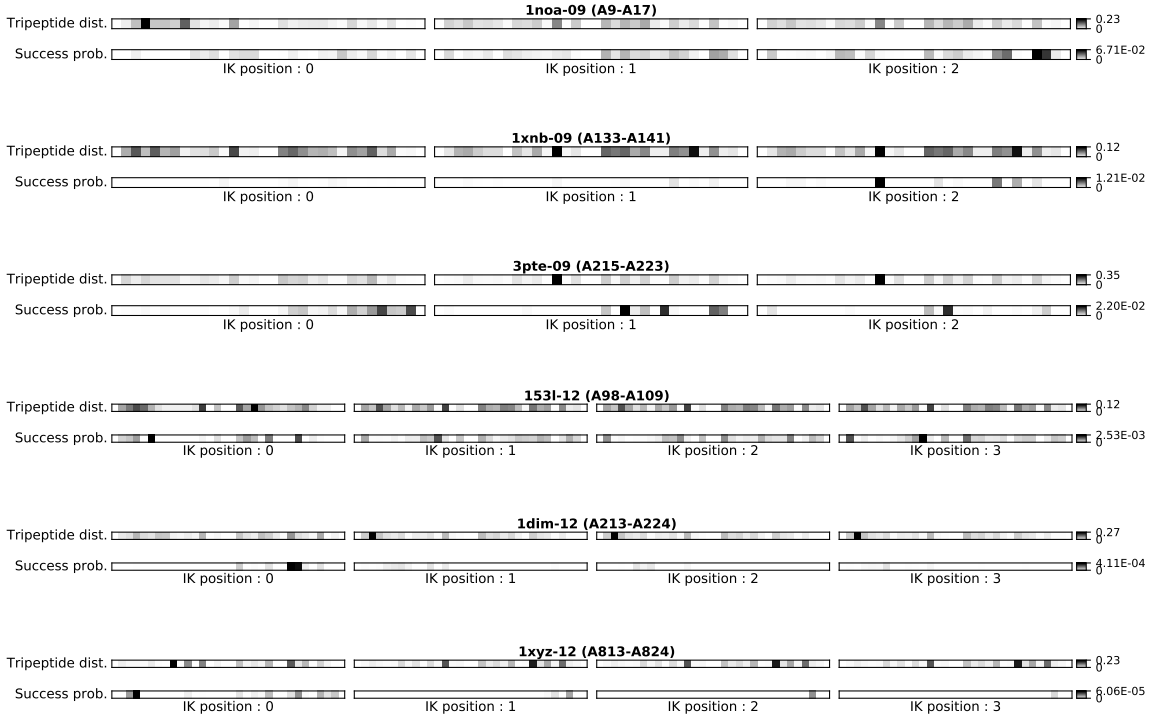


Fig. S17((e)): Quaternion and length

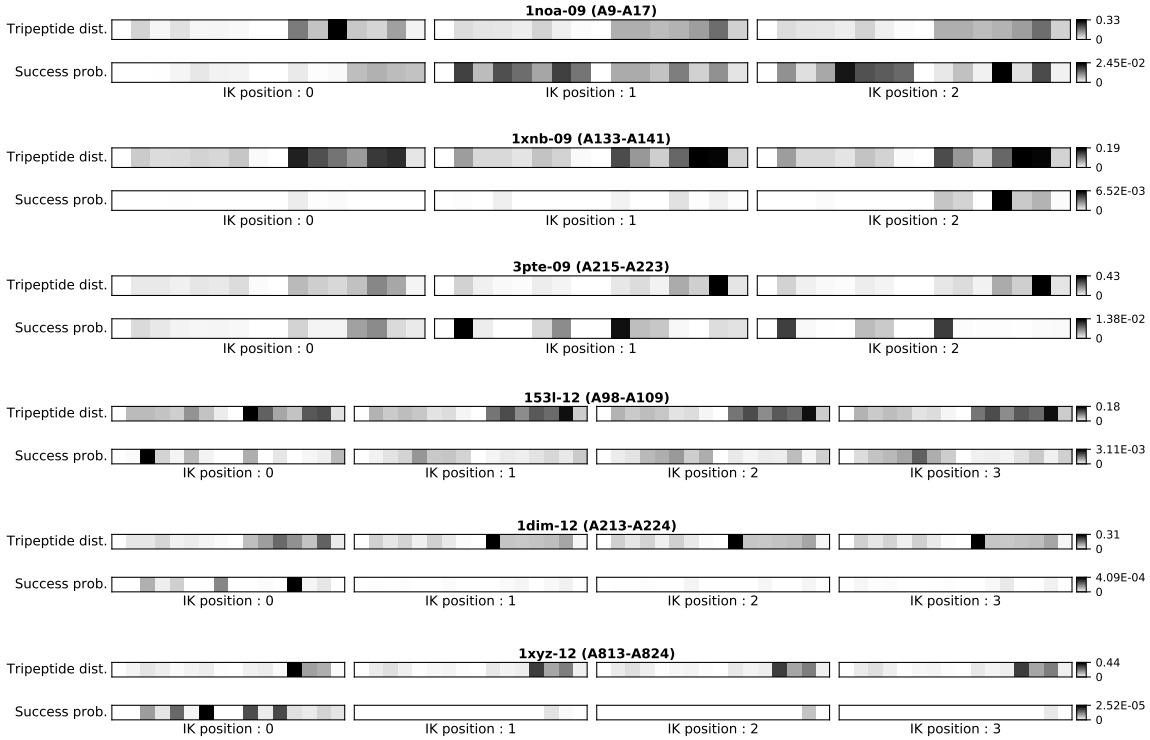


Fig. S17((f)): Axis-angle

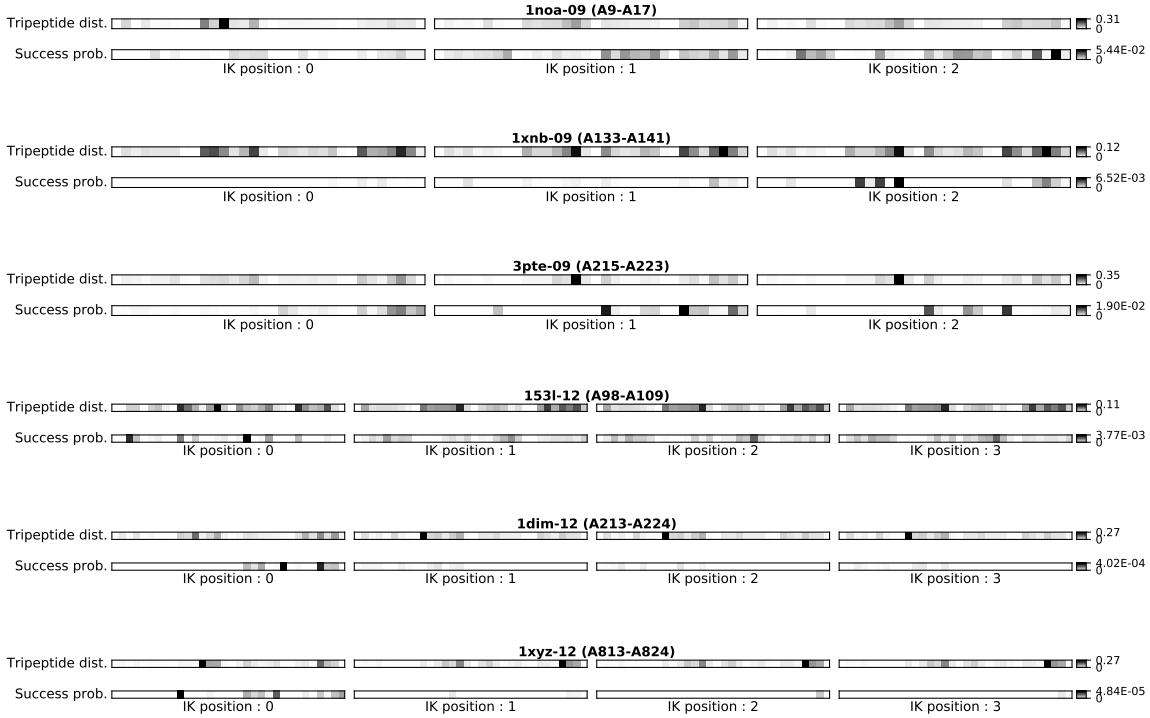


Fig. S17((g)): Axis-angle and length