JCTC Journal of Chemical Theory and Computation

Article

pubs.acs.org/JCTC

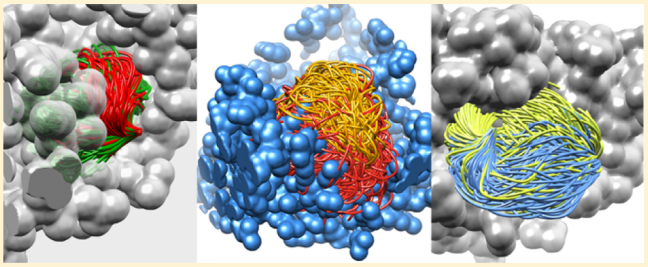
Random Coordinate Descent with Spinor-matrices and Geometric Filters for Efficient Loop Closure

Pieter Chys* and Pablo Chacón*

Structural Bioinformatics Group, Biological Chemical Physics Department, Institute of Physical Chemistry Rocasolano (IQFR), Consejo Superior de Investigaciones Científicas (CSIC), Calle de Serrano 119, Madrid 28006, Spain

Supporting Information

ABSTRACT: Protein loop closure constitutes a critical step in loop and protein modeling whereby geometrically feasible loops must be found between two given anchor residues. Here, a new analytic/iterative algorithm denoted random coordinate descent (RCD) to perform protein loop closure is described. The algorithm solves loop closure through minimization as in cyclic coordinate descent but selects bonds for optimization randomly, updates loop conformations by spinor-matrices, performs loop closure in both chain directions, and uses a set of geometric filters to yield efficient conformational sampling. Geometric filters allow one to detect clashes and constrain dihedral angles on the fly. The RCD algorithm is at least comparable to state of the art loop closure algorithms due to an excellent balance between efficiency and intrinsic sampling capability. Furthermore, its efficiency allows one to improve conformational sampling by increasing the sampling number without much penalty. Overall, RCD turns out to be accurate, fast, robust, and applicable over a wide range of loop lengths. Because of the versatility of RCD, it is a solid alternative for integration with current loop modeling strategies.



1. INTRODUCTION

Loop closure is the problem of finding geometrically feasible conformations for a short peptide segment between two fixed amino acid residues in a protein. This problem originates in the existence of low-resolution regions of electron density maps constructed from crystallographic X-ray data in the neighborhood of protein loops.[1] Loop closure has also been investigated to a certain extent in ring closure studies[2,3] and in the robotics field where the inverse kinematics problem is very similar.[4,5] Protein loop closure is very important in loop modeling and ab initio studies where the ultimate goal is to predict the most energetically favorable loops given the local protein environment.[6−14] Typically, based on sequence homology modeling, regular secondary structures are assigned to an unknown protein after which the remaining variable regions are suitable for loop modeling.[14,15] Loop closure constitutes then the first critical step of the complete loop modeling and aims at obtaining a conformational loop ensemble satisfying the geometrical closure constraints.

For loop closure, a loop must connect with a suitable backbone conformation the amino-terminal fixed residue with the carboxy-terminal target residue (see Figure 1). The amino-terminal and carboxy-terminal anchor residue are also simply referred to as N- and C-anchor. In principle, the goal is to have a perfect fit at both residue anchors or at least a very small root-mean-square deviation (RMSD) between the loop anchors and their target positions. The exact loop length has some relevance in the selection of a suitable loop closure method. Loop sizes typically range from 4 to 12 residues,[15−17] but larger sizes up to 17 residues have also been examined.[18]

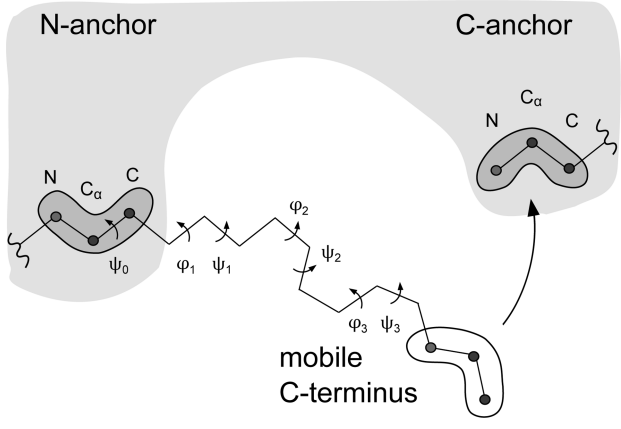


Figure 1. Loop closure: the oligopeptide must connect the amino-terminal anchor residue (N-anchor) with the carboxy-terminal anchor residue (C-anchor). The mobile carboxy-terminal residue (C-terminus) must reach its target position by changing the dihedral angles φ and ψ along the loop backbone.

To solve the loop closure problem, different methods are available,[3,15,16,19−22] and the distinction with loop modeling algorithms is also not sharp.[11,18,23−28] Pure loop closure methods can be classified as ab initio methods (1), homology

Received: November 7, 2012
Published: February 15, 2013

methods (2), or hybrids (3) between them. Ab initio methods generate ensembles of loop conformations from scratch, whereas homology methods use fragment libraries to construct loops with suitable peptide fragments. Hybrid methodologies intend to use the best of both worlds and are both computation- and library-driven.[22] Methodologically, loop closure methods can also be classified in the following three classes:[22] analytical methods (1), build-up methods (2), and iterative methods (3). Build-up methods (e.g., loop closure routine in PLOP) construct loops residue by residue and subsequently refine these structures. Analytical approaches calculate exact solutions and are fast.[20,29,30] The main analytical technique is the polynomial resultant method which solves analytically triaxial loop closure.[20] Larger loops are in essence solved by splitting the complete loop into three large fragments and applying the triaxial loop closure procedure. Although the polynomial resultant method by virtue of its exactness and raw speed appears the best choice for loop closure, it lacks some of the flexibility of the iterative methods. Iterative methods either employ local minimization techniques with an exact molecular geometry or use multidimensional minimization allowing geometrical distortions. One-dimensional minimization is used in the Cyclic Coordinate Descent method (CCD) to superimpose mobile and target anchors by finding the optimal dihedral angle for the current rotation bond.[15,16] The CCD method is more an analytical/iterative method than a purely iterative or analytical method. Multidimensional minimization is done by Lagrange multipliers in the random tweak method,[3] and here all dihedrals are changed at the same time for a single iteration. Random Tweak (RT) is classified as a purely iterative method. Direct tweak is a further development of RT and automatically incorporates avoidance of steric clashes in the minimization step.[21] The polynomial resultant, RT, and CCD methods do not include clash avoidance during actual loop closure.

Cyclic coordinate descent is in essence one of the most simple loop closure methods with both advantages and disadvantages as compared to other algorithms. Probably, robustness and algorithmic flexibility are its main assets as compared to its competitors. From the iterative methods, RT appears to be much faster in comparative tests from ref 14. In loop closure without clash detection the polynomial resultant yields exact solutions and is also faster than CCD. See ref 22 for a recent review of loop closure methods in the context of loop simulations.

Here, a new analytical/iterative algorithm is presented to perform more efficiently loop closure. The algorithm, denoted random coordinate descent (shortly RCD), combines algorithmic principles from CCD algorithms[15−17] with distinct and new features. Rotation bonds are chosen randomly instead of sequentially, and a hybrid spinor-matrix approach is implemented for fast conformational updating. Spinor-matrices should in principle yield the fastest computational scheme.[31,32] For the optimization protocol, a semianalytical procedure is used and is based on theory in ref 15. A further key element is the introduction of different types of geometrical filters with specific code placement in the algorithm. Included geometric filters are clash detection of the loop with the protein surrounding (1), collision detection between the loop backbone atoms (2), and filters constraining the dihedral angles to Ramachandran ranges (3). Experimental and comparative tests are made about these geometric filters in RCD, their relative code placement, and the repercussions on sampling and time performance for loop closure. It is shown that a versatile, efficient, and robust algorithm is obtained which can be set up for sampling and/or time performance.

2. METHODS

The main scheme of the RCD algorithm is shown in Figure 2. The inputs of the algorithm are the protein coordinates and the

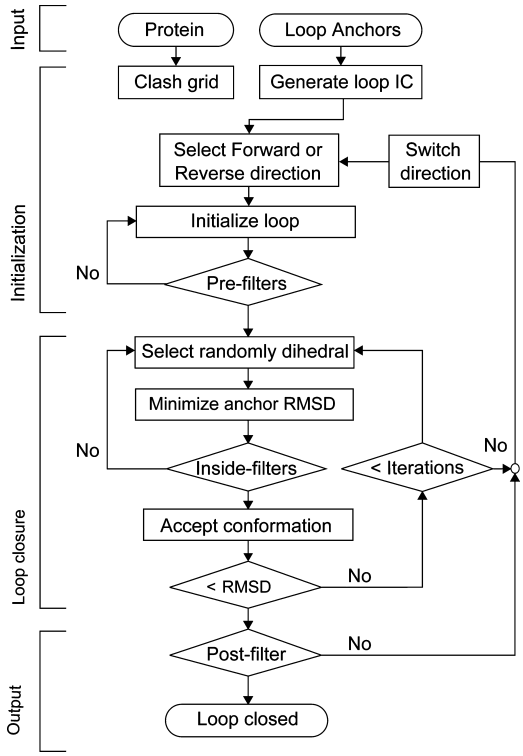


Figure 2. Algorithmic scheme of random coordinate descent (RCD) for generating a single loop solution.

two anchor positions of the N- and C-terminal end residues of the target loop. The protein coordinates are used to construct a 3D grid to screen out loops having steric clashes with the local protein environment, whereas the anchors are used as starting points to build an initial and open loop candidate. To obtain a loop candidate, the corresponding internal coordinates are generated from a peptide backbone template or are directly taken from the native loop conformation. As explained later, the native internal coordinates were only used for validation purposes. Then, from these internal coordinates an initial conformation is generated with the N-anchor in its native position but the C-anchor being randomly out of place. With the geometric pre-filters active, this open loop candidate has no clashes with the local protein environment and/or conforms the energetically favorable regions of the Ramachandran plot. The actual loop closure procedure on the 3D grid can now start by keeping only the N-anchor fixed and allowing the C-anchor to freely move. In each iteration of the algorithm, a dihedral angle is randomly selected and minimized to obtain the most optimal rotation that reduces the RMSD between the mobile and target C-anchor. Geometric inside filters now determine whether the loop conforms the Ramachandran valid regions and/or whether the internally updated loop conformation does not clash with the protein loop surrounding. If the loop passes these inside-filter checks, the updated loop conformation is

accepted. If it is rejected, the old conformation is reused. In either case, the cycle starts over again by selecting randomly a new and different dihedral angle.

This loop closure protocol iterates until the anchor RMSD becomes lower than a given threshold (e.g 0.25 Å as in ref 14). If now the converged loop does not pass the post-filter checks for intraloop backbone clashes, the algorithm basically restarts a new initialization. This means that a new and open loop candidate to perform loop closure is generated without reference to the post-filter rejected loop. Similarly, if the mobile anchor has not converged to its target position in less than a fixed number of iterations the initialization procedure is also triggered and executed to yield a new loop candidate. On the contrary, if the converged loop candidate passes the post-filter checks, it is accepted as a loop closure solution. The full algorithm is then repeated from a new, different initial candidate to obtain the desired number of closed loop solutions.

In addition, the overall algorithm allows one to apply the complete loop closure in the reverse chain direction during which the C-anchor becomes fixed and the N-anchor mobile. In a standard RCD run where thousands of closed loops are generated, loop closures will be performed in a single sense but alternating between forward or backward directions in separate runs. The ability to close loop candidates in both senses enhanced sampling variability. Moreover, we will see that switching the direction whenever the current loop fails to converge improved time performance. In this case, the most favorable loop closure direction naturally dominates the loop closure.

The RCD algorithm consists now of four functional units: a scheme to select rotation bonds (1), a dihedral angle optimization routine (2), a procedure to update loop conformations (3), and a geometric filter part (4). The bond selection scheme picks always randomly a bond while avoiding the previously chosen bond. It is a simple scheme, and we will not discuss it further. The other functional units are explained chronologically in the next three sections (sections 2.1−2.3). Afterward, technical details are presented (section 2.4), after which the method validation for RCD will be highlighted (section 2.5).

2.1. Dihedral Angle Optimization. When the random bond selection scheme in the algorithm has picked a rotation bond, the minimization routine must now calculate the optimal rotation angle such that the mobile C-anchor moves as closely as possible to the target anchor position. This is equivalent to minimization of the root-mean-square-deviation between the mobile and target anchor atom coordinates. We employ a similar approach as in ref 15 but use the equation with dot products [eq 8 in ref 15] instead of the final and analytical equation derived there. The routine is thus a two-step scheme which is now commented upon in detail.

In the first step, the protein loop is aligned in such a way that the selected bond is parallel to the x axis. The unit bond vector becomes then equal to $[1\ 0\ 0]^T$. To that purpose, the rotation matrix A is computed, which obeys

$$[100]^T = A[b_x b_y b_z]^T \tag{1}$$

with (b_x,b_y,b_z) being the normalized bond coordinates prior to alignment. Matrix multiplication of A with both the mobile and target anchor coordinates relative to the first atom of the chosen bond results in the transformed coordinates ($\{\mathbf{m}_i\}$ and $\{\mathbf{a}_i\}$ for i = 1, 2, 3). In the second step, the optimal angle δ is now calculated:[15]

$$\delta = \arctan\left(\frac{\sum_{i=1}^{3} \mathbf{m}_i \cdot |\mathbf{a}_i| \hat{\mathbf{a}}_{i\perp}}{\sum_{i=1}^{3} \mathbf{m}_i \cdot \mathbf{a}_i}\right) \tag{2}$$

with $|\mathbf{a}_i|$ being the magnitude of $\mathbf{a}_i$ and $\hat{\mathbf{a}}_{i\perp}$ being the orthonormal vectors to the planes formed by $\mathbf{a}_i$ and $[1\ 0\ 0]^T$. In eq 2 only the y and z coordinates from $\{\hat{\mathbf{a}}_{i\perp}\}$ and $\{\mathbf{m}_i\}$ must be used in the numerator since the alignment of the rotation bond makes the x components for $\{\hat{\mathbf{a}}_{i\perp}\}$ naturally zero.

2.2. Conformational Updating. Loop conformations are updated in the loop closure routine by using spinors and matrices, and this is done right after the minimization step. The terminal loop segment after the chosen rotation bond is rotated through the computed dihedral angle. Spinors are rotational operators from geometric algebra (GA) and are isomorphic to quaternions.[33−36] First, a base spinor R is constructed:

$$R = \cos\left(\frac{\delta}{2}\right) + \sin\left(\frac{\delta}{2}\right)\imath\hat{\mathbf{b}} \tag{3}$$

with δ being the rotation angle obtained from dihedral optimization, $\hat{\mathbf{b}}$ being the normalized rotation axis vector (unit bond vector), and $\imath$ being the pseudoscalar, which is an entity in GA to manipulate expressions. The vector $\hat{\mathbf{b}}$ can always be computed from the existing loop conformation. The computed spinor R corresponds now to the 4-tuple (α,b_x,b_y,b_z) and is used as input for the second step. Herein, the rotation matrix A is computed:

$$A = 2\begin{bmatrix} \alpha^2 + b_x^2 - \frac{1}{2} & b_x b_y - \alpha b_z & b_x b_z - \alpha b_y \\ b_x b_y + \alpha b_z & \alpha^2 + b_y^2 - \frac{1}{2} & b_y b_z - \alpha b_x \\ b_x b_z - \alpha b_y & b_y b_z + \alpha b_x & \alpha^2 + b_z^2 - \frac{1}{2} \end{bmatrix} \tag{4}$$

If now $\mathbf{x}_i$ are the current coordinates of loop atom i and $\mathbf{x}_e$ is the coordinate of the end atom of the selected rotation bond, the updated coordinates $\mathbf{x}_i'$ become

$$\mathbf{x}_i' = A(\mathbf{x}_i - \mathbf{x}_e) + \mathbf{x}_e \tag{5}$$

and eq 5 must be applied to all atoms located in the carboxy- or amino-terminal direction of the chosen rotation bond. The set of eqs 3, 4, and 5 constitutes the two-step hybrid spinor-matrix scheme for updating coordinates. It corresponds with the quaternion-matrix scheme but is derived in the broader framework of geometric algebra. Results on the equivalent quaternion-matrix scheme show it to be one of the fastest available methods.[31,32]

2.3. Geometric Filters. The filter part of the algorithm consists in fact of three different types of geometric filters. As mentioned before, they are integrated at different places in the algorithm: pre-, inside- and post-filter placement versus the closure routine (see Figure 2). We now highlight them separately.

2.3.1. Grid Clash Filter. To check for steric clashes between the loop and its local protein surrounding, a simple three-dimensional grid is used. Grid points are assigned 1 if they belong spatially to the protein surrounding and 0 if space is available to place loop atoms. This is done by constructing van der Waals spheres at the protein atom positions. Loop atoms

can then be checked for steric clashes with the protein atoms by making a boolean check at their grid positions. This filter can be activated as pre-, inside-, or post-filter.

2.3.2. Loop Clash Filter. This filter checks for inside clashes between all loop backbone atoms and calculates the interatomic distances to that purpose. A cutoff factor for atomic softness was applied and always multiplied with the sum of the two atomic radii involved in the computation. The cutoff was set at 0.5 in agreement with ref 14, and radius values were taken from ref 37. This geometric filter was implemented both as an inside filter and a post-filter.

2.3.3. Adaptative Ramachandran Filter. This filter restricts the dihedral angles φ and ψ according to valid domains of the Ramachandran plot. One filter can function during initialization (pre-filter) and another one works inside the loop closure (inside filter) but before the grid clash filter is applied (see Figure 2). We followed a very basic procedure in which the peptide dihedral angles are restricted to simplified and rectangular areas in the Ramachandran plot. Here, φ angles are restricted to $[-175°,-40°]$ and ψ angles to the range $[-60°,175°]$ as was defined in ref 16. The ω angles were always kept fixed. Two exceptions are made to the application of a Ramachandran filter. First, proline has a fixed φ, and second, glycines retain complete dihedral freedom. The initialization filter simply rejects or accepts a loop conformation, but the inside filter is more subtle. If the optimal angle is out of the correct Ramachandran range, the filter calculates the rotation that is still possible within the valid range. The loop is accordingly updated. However, when this angle becomes too small ($<|6 \times 10^{-5}|°$), the rotation is rejected and the Ramachandran range restriction is taken away for this dihedral during the remainder of the current loop closure. Such a filter is denoted here as an adaptative Ramachandran filter. We experimented with several other Ramachandran base filters,[15,17] but these did not have any effect, which is in agreement with the observations in refs 15 and 17.

2.4. Technical Details. *2.4.1. Initialization.* To randomize the loop conformation the existing or created dihedrals are by default cycled several times by randomly rotating the bonds. If a pre-filter is active, it checks the conformation always right after this step and repeats the protocol if needed. The available degrees of freedom to attain loop closure are as follows. Each peptide residue between the anchors can rotate both its φ and ψ angles, but the corresponding ω angle is always kept fixed. The ψ angle of the amino-terminal N-anchor and the φ angle of the carboxy-terminal C-anchor are also allowed to rotate. These degrees of freedom also apply when closing the loop in the reverse chain direction. One exception is made to these degrees of freedom. The proline residue gets assigned a fixed φ angle in correspondence with its rigid nature. Overall, for an n-residue loop, $2(n-2)+2 = 2n-2$ rotational degrees of freedom are present (excluding proline deviations).

2.4.2. Grid Clash Filter. A cubic and orthogonal grid is always constructed on the basis of the end atom positions of the loop. The most important grid characteristic is the step size (Å), which is the distance between two grid points along its orthogonal axis directions. We found optimum values for the step size around 0.20−0.25 Å. A single radius value of 3.0 Å was used for all atom types (double of 1.5 Å). To account for softness in clashes, a softness factor of 0.5 was used in accordance with ref 14.

2.4.3. Implementation. For the implementation, we programmed in C++, and computations were done in Linux (centOS 5.5) on a 64-bit machine. An Intel Core (TM) i7 CPU 950@3.07 GHz processor with 12 GB of RAM was used. The executables were compiled with the Intel (version 12.1) compiler under -O3 compilation level. The RCD algorithm as a tool for doing protein loop closure is available at http://chaconlab.org/rcd and comes with a concise manual.

2.5. Method Validation. To validate our algorithm, we followed the three-step procedure depicted in Figure 3. First,

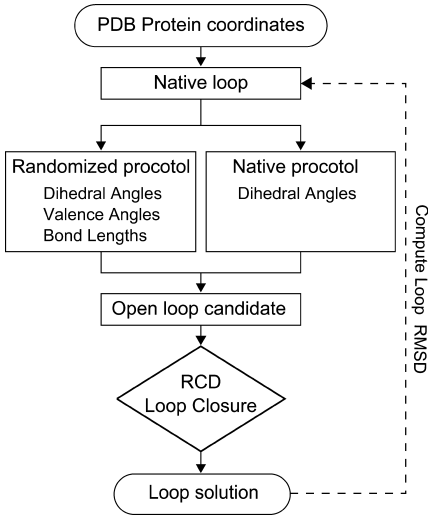


Figure 3. Scheme illustrating the method validation used for testing the RCD algorithm.

open loop candidates are generated from loop benchmark sets. Second, RCD performs loop closure as explained before. Finally, RMSD values between the loop solutions and the original native coordinates are calculated to evaluate the quality of the loop closure.

2.5.1. Loop Candidate Generation. To start, native and closed loops were extracted from known loop structures to obtain the exact loop atom coordinates. We generate random initial conformations by keeping the loop anchored at one terminus and moving away the other terminus from its proper location. The mobile terminus is moved by randomly perturbing the closed conformation through rotation of its dihedral angles. The resulting loop is then a loop candidate for loop closure, meaning that the RCD algorithm must restore the perturbed and mobile terminus to its original position. This is the basic scheme, but we employ two variants depending on the internal coordinate sets used. In the most simple case that we call the native protocol, bond length and valence angle values are kept from the native conformation, and only dihedrals are allowed to vary. This is the standard perturbation protocol used to generate validation tests in line with previous loop closure algorithms (see for instance refs 14 and 15). Thus, the native protocol is used for comparative purposes.

To validate the RCD algorithm in the algorithmic tests which we present later, we were more stringent and varied as well bond lengths and valence angles. This second protocol, denoted randomized, corresponds to the real loop closure case in which only the terminal anchors are given as part of the problem statement. In that case, the internal coordinates are of course introduced by a set of default values. More exactly for the randomized protocol, bond lengths and valence angles for the backbone geometry were taken from ref 38 and small random deviations added by using ranges of refs 38 and 39.

Table 1. Comparison of RMSD Measures and Timings for Test Runs with Different Grid Clash Filter Combinations Set up in the RCD Algorithm[a]

	loop size											
	4			8			12			15		
	RMSD (Å)			RMSD (Å)			RMSD (Å)			RMSD (Å)		
clash filter	avg	min	t(min)	avg	min	t(min)	avg	min	t(min)	avg	min	t(min)
No	1.30	0.34	0.17	3.82	1.14	0.15	6.38	2.10	0.20	8.86	2.76	0.54
Inside	1.29	0.33	0.21	3.46	1.07	0.42	5.42	1.92	1.67	7.12	2.53	2.85
Post	1.29	0.33	0.23	3.46	1.20	0.36	5.04	1.85	9.37	6.88	2.42	6.40
Inside + Loop	1.29	0.33	0.31	3.45	1.14	0.84	5.41	1.92	2.82	7.21	2.47	6.02

[a]Values apply to 5000 randomized loop closure solutions (RMSD anchor threshold 0.25 Å) with benchmark sets from refs 5 and 15. The grid clash filter designations are as follows. No refers to the algorithm without a filter active. Inside corresponds to the inside-filter version and post to post-filter placement. The Inside + Loop case includes both the grid and loop clash filters as inside filters. Abbreviations: avg = average RMSD over all loop solutions, min = average over the lowest obtained RMSD values taken from each loop in the benchmark, t = time.

Moreover, to improve the conformational screening for ideal loops, the random deviations on internal coordinates are performed each time a new closure is started. Thus, each loop candidate has a completely distinct set of internal coordinates. Finally, whether using the native or randomized protocol, the terminal anchors always retained their native or given internal coordinates in all validation tests to allow in principle for a perfect fit upon anchor convergence.

2.5.2. Benchmark Sets. Two standard benchmark sets were used. The first benchmark set is a classical set of protein loops investigated in refs 15−17 and consists of 10 loops for three peptide sizes: tetra-, octa-, and dodecapeptides. We added 10 loops of 15-peptides from ref 5 to have representatives for the class of long loops. We refer later to this set as benchmark set 1. The second benchmark set consists of 51 octapeptides, 17 undecapeptides, and 10 dodecapeptides and stems from ref 14. We needed to remove three octapeptides from the original 54 octapeptides since we found that they were in fact non-apeptides [1awd.pdb (55−63), 1byb.pdb (246−254), and 1ptf.pdb (10−18)]. This is expected to have negligible influence on the obtained results since the octapeptide group is large ($n \simeq 50$). This resulting set of protein loops is later referred to as benchmark set 2. All PDB files were retrieved from the PDB database, individually checked and prepared for subsequent use. More details about the benchmark sets can be found in the Supporting Information. In the tests, 5000 loop solutions were always generated per PDB entry from these benchmark sets with a RMSD anchor threshold of 0.25 Å. This anchor threshold applies to the two-atom approach as used in ref 14, and it is the two last atoms of the terminal anchor residue which are always used.

2.5.3. RMSD Validation. After actual loop closure, our validation protocol consisted of measuring the RMSD distance between the backbone atoms (including oxygen atoms) of each native loop and the obtained loop solutions. In such a manner, one can have an estimation of the sampling power of the method. We employed two statistical parameters. Parameter $\langle \mathrm{RMSD} \rangle_{\mathrm{min}}$ or RMSD min is the average RMSD value for the loop solutions closest to each native loop over a benchmark set. Analoguously, $\langle \mathrm{RMSD} \rangle_{\mathrm{avg}}$ or RMSD $_{\mathrm{avg}}$ refers to the average value of the RMSD averages per loop entry. Stated differently, each PDB loop is characterized by a RMSD value for its solution ensemble, and averaging this value over the benchmark yields $\langle \mathrm{RMSD} \rangle_{\mathrm{avg}}$. We naturally assume that $\langle \mathrm{RMSD} \rangle_{\mathrm{min}}$ values indicate the potential for absolute conformational sampling, that is, to get close to the native loop. On the other hand, $\langle \mathrm{RMSD} \rangle_{\mathrm{avg}}$ likely indicates how well the whole loop ensemble lies around the native loop conformation. In general, $\langle \mathrm{RMSD} \rangle_{\mathrm{min}}$ and RMSD avg correlate well and positively in the test results. We observed as well that the standard deviations for these sampling indicators are large. However, this appears to stem from the fact that individual loops with given length show large variation in both RMSD minima and averages. For a benchmark set there can thus be quite some heterogeneity in terms of sampling statistics. Nevertheless, repeating an experiment several times (see Results, section 3.2) showed that the spread between separate runs over the same benchmark was much lower and that $\langle \mathrm{RMSD} \rangle_{\mathrm{min}}$ and $\langle \mathrm{RMSD} \rangle_{\mathrm{avg}}$ are therefore accurate values for the sampling characteristics in our experiments.

3. RESULTS

3.1. Algorithm Tests. To detect an optimal setup for RCD, we examine how geometrical filters affect the sampling and time performance. Both type and placement of geometrical filters in the RCD algorithm are considered in two computational tests.

Table 1 shows experimental results of a first test with RCD algorithms having different clash filter combinations. Using the randomized protocol, 5000 loop solutions were generated for benchmark set 1 (see section 2.5.2). These are also the test conditions for the second test presented later. The RCD Inside and RCD Post versions have the grid clash filter coded respectively inside and after the loop closure routine. RCD Inside + Loop includes the intraloop clash filter located inside and at the end of the loop closure routine. The results obtained with the smallest loops were very similar in all cases. However, for medium and long loops, all versions with grid clash filter active (Inside, Post, and Inside + Loop) have lower RMSD values than the RCD version without the grid filter (No). The RMSD averages illustrate the improved conformational sampling best. The timings for the post-filter version increase significantly versus the inside-filter implementation, and the latter approach is thus advantageous for time performance. Remark that the post-filter layout corresponds to what most closure methods use as clash detection protocol in the study.[14] It also appears from Table 1 that post-filter placement might offer some benefit in terms of conformational sampling for big loops. When the loop clash filter is placed inside the loop closure routine of the RCD Inside variant (Inside + Loop), time performance decreases without any benefit in terms of sampling performance. For absolute time performance, the inside-filter setup does perform well with respect to the No grid

Table 2. Comparison of Test Runs of the RCD Algorithm with Different Ramachandran Filter Schemes[a]

	loop size											
	4			8			12			15		
	RMSD (Å)			RMSD (Å)			RMSD (Å)			RMSD (Å)		
Ramach. filter	avg	min	t(min)	avg	min	t(min)	avg	min	t(min)	avg	min	t(min)
No	1.29	0.33	0.21	3.46	1.07	0.42	5.42	1.92	1.67	7.12	2.53	2.85
Pre	1.08	0.31	0.16	3.39	0.95	0.34	5.23	1.68	0.91	7.23	2.52	1.84
Pre + Inside	0.87	0.33	0.49	3.17	0.89	0.87	4.87	1.58	3.12	7.02	2.46	4.88
Swithcing	0.97	0.28	0.49	3.04	0.81	0.77	4.93	1.59	3.04	6.80	2.36	2.06

[a]Filter designations are as follows. The No case uses no Ramachandran filter but has the grid clash inside filter active. The Pre form adds the Ramachandran filter as a pre-filter. The Pre + Inside setup uses both the Ramachandran pre- and inside filter while retaining the grid clash filter. Switching refers to the pre + inside mode applied in the forward and backward protein directions and changing direction upon convergence failure in one direction. See for more details Table 1.

setup despite the observed time deficit. Only for 12 peptides for instance do the computations slow down roughly by a factor 10. This time performance is all the better if one considers that the timings are better than the fastest reported method in ref 14, RT. For tetrapeptides, the results differ from above, and here sampling is nearly equal for all versions as shown by the $\langle RMSD \rangle_{min}$ and $\langle RMSD \rangle_{avg}$ values. The time performance of post- and inside-filter setup here is comparable, and both other RCD versions do not show very different timings. Overall, Table 1 indicates that the Inside variant has the best sampling and time characteristics.

Table 2 shows results of a second test with Ramachandran filters and the bidirectional loop closure protocol added on top of the optimal RCD variant from Table 1. The reference in Table 2 (No) therefore corresponds to the best RCD layout from Table 1 (Inside). Here, the Ramachandran pre-filter approach (Pre) appears to speed up the loop closure in longer loops and samples at least equally well as the reference. RCD with both the pre- and inside Ramachandran filters active (Pre + Inside) samples however significantly better than the reference, as is indicated by the lower $\langle RMSD \rangle_{min}$ and $\langle RMSD \rangle_{avg}$ values. This version has also better RMSD values than the pure pre-filter layout, suggesting that the double filter combination screens intrinsically better. Concurrently, it is also evident in Table 2 that the improved conformational sampling comes at the expense of time performance. Finally, the application of both Ramachandran pre- and inside filters together with the bidirectional protocol (Switching) seems to yield the best overall performance but is closely matched with only applying RCD in the forward chain direction (Pre + Inside).

The overall results of the two tests with RCD are nicely summarized and illustrated in Figure 4. Herein, RMSD distributions of the loop solution ensemble for protein loop 1i0h.pdb (residues 145−152) are shown for three variants of RCD. The green RMSD distribution corresponds to a loop solution ensemble of RCD without a grid clash filter. It is now observed that adding a grid clash inside filter (blue) shifts the distribution without a grid clash filter (green) to the left and toward lower RMSD values. Adding as well Ramachandran filters (red) shifts the blue RMSD distribution further to the left. The observed shifts are valid over the full RMSD range, indicating a shift of the loop solution ensembles themselves, and this observation correlates well with the experimental changes in $\langle RMSD \rangle_{avg}$ and $\langle RMSD \rangle_{min}$ values. As such, it is seen how successive inclusion of geometric and Ramachandran filters on top of a core RCD algorithm without filters results in

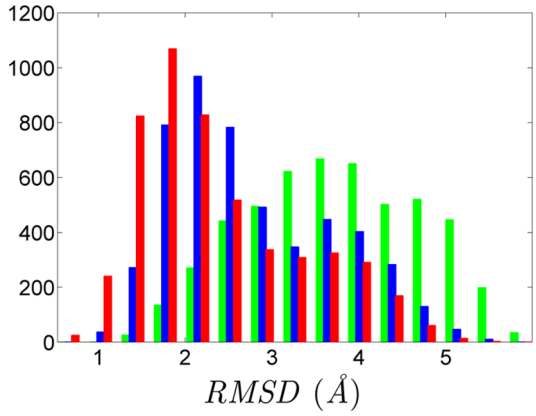


Figure 4. Illustrative RMSD distributions with respect to the native loop for the eight-peptide loop solution ensemble of 1i0h (residues 145−152). Green is RCD without a grid clash filter, blue is RCD with grid clash filter, and red is RCD with grid clash and Ramachandran inside filter.

improved sampling characteristics. Figure 5 shows in conjunction how the red and green sampling distributions from Figure 4 translate to actual loop solutions in the local protein environment.

The previous test results with randomized internal coordinates mimic the more realistic loop closure case in which one has no prior knowledge about the ideal loop solution. However, for comparative reasons, the same experimental runs can be done using the native protocol. These tests yield results in agreement with those obtained from the randomized protocol (see the Supporting Information). Timings are in general slightly faster, and conformational sampling is nearly identical for octa-, dodeca-, and pentadeca-peptide loops. Only tetrapeptide loops show some deviations. Corresponding $\langle RMSD \rangle_{min}$ values to those in Tables 1 and 2 for tetrapeptides are around 0.15 Å, which is approximately half of the values seen with randomized coordinates. For the remainder, the results with 4-peptides agree with the observations in larger loops, and here a more systematic trend in the timings can be observed.

3.2. Comparative Tests. This section compares the RCD algorithm with other loop closure algorithms and is split up in two parts. First, a concise comparison is made with cyclic coordinate descent (CCD) algorithms which do not take into account clashes with the local protein surrounding. Second, a comparison is made with other loop closure methods which effectively yield clash free loop solutions. Experimental runs

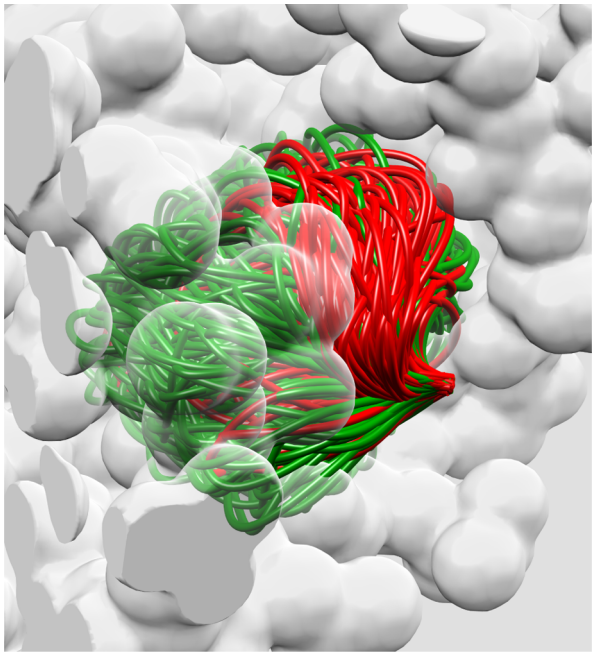


Figure 5. Conformational backbone sampling with (red) and without (green) geometric filters. Protein loop 1i0h.pdb (8-peptide, 145−152) is closed 1000 times with RMSD threshold 0.25 Å.

with RCD were always performed under identical conditions of the relevant loop closure studies from which the experimental data were taken.[14,15]

Cyclic coordinate descent algorithms were taken as a basis to develop the RCD algorithm, and especially the original CCD algorithm[15] must be mentioned in this regard. The other algorithms[16,17] take on new development lines and are in essence already successors intending to supersede the original CCD algorithm. In fact, a critical statement made in ref 15 suggested including clash filters inside a CCD-derived algorithm and evaluating the possible performance effects. This development line was partially followed here. Nevertheless, RCD can also be compared with CCD when all the geometric filters are switched off since in the original CCD no clash filters are present. Such a test allows assessment of the pure random coordinate descent principle versus the cyclic coordinate descent principle. Here, RCD runs done under identical conditions without geometrical filters for 4-, 8-, and 12-peptides[15] yield $\langle RMSD \rangle_{min}$ = 0.28, 1.16, and 2.10 Å, whereas the original CCD paper[15] reports 0.56, 1.59, and 3.04 Å, respectively. These results were produced with randomized internal coordinates. The relevant benchmark set is actually benchmark set 1 but without the 15-peptides we introduced for the algorithm tests. We find thus that even without geometric filters active, a random coordinate descent algorithm improves upon a basic CCD algorithm. Timings (3.9−7.1 ms/closure) indicate excellent relative performance (23−37 ms/closure[15]) but probably also result from improvements in processor technology. Remarkably, time performance corresponds well with the optimum performance observed in the successor Full Cyclic Coordinate Descent (FCCD)[17] (4.5−7.1 ms/closure), which, although not completely comparable with RCD in terms of geometry due to its simplified C_α model, does use also two degrees of freedom per loop unit (C_α atom for FCCD). Thus, the RCD algorithm improves significantly upon the original CCD when no geometric filters are used. It likely indicates that a random coordinate descent algorithm screens better loop solutions than a cyclic coordinate descent algorithm even if the local protein surrounding is not considered.

Second, we compare RCD with other loop closure methods that yield clash free loop solutions. To that purpose, relevant data from ref 14 are compiled in Table 3 together with experimental results here. Native internal coordinates were used in agreement with ref 14 and run with the best RCD algorithm from Table 2 (Switching). This optimal RCD algorithm includes Ramachandran pre- and inside filters, an inside grid clash filter, intraloop clash post-filter, and a switching direction mechanism upon failure. For this comparison, benchmark set 2 is used. This RCD algorithm at an identical sampling number yields averaged RMSD minima of 0.78, 1.38, and 1.60 Å for 8-, 11-, and 12-peptides which are only second to that of the best method (direct tweak). Remark that direct tweak is the only method that also uses clash detection inside its loop closure routine from all reported methods in ref 14. RCD outperforms the examined CCD implementation from ref 14, and this is a CCD version that has a post-filter implemented to eliminate steric clashes with the protein surrounding. Overall, the RCD algorithm samples loop solutions extremely well based on Table 3, and the time performance is also outstanding. The speed-up compared to reported direct tweak values ranges from 6- to 17-fold. Furthermore, we found that when using RCD without Ramachandran inside filters (denoted RCD Pre), RMSD values were still only second to those of direct tweak but with the advantage of larger speed-ups (25−60-fold). It is unlikely that these speed-up differences only result from advances in processor technology. In that case, RCD outperforms the other methods, and only direct tweak is a match due to its better intrinsic sampling capability at a given sampling number (number of closures/loop).

Observe now that in the comparison of Table 3 we strictly followed the test conditions of ref 14. But, since the RCD algorithm is fast, we considered it useful to test the effect of increasing the sampling number n up to a value 20 times more.

Table 3. Comparison of RMSD Minima from RCD Runs with Literature Results[14a]

	loop size		
	8	11	12
algorithm	RMSD (Å)	RMSD (Å)	RMSD (Å)
random tweak	1.22	2.22	2.64
CCD	1.20	2.11	2.57
wriggling	1.43	2.24	2.68
PLOP-build	0.99	2.18	2.69
direct tweak	0.69	1.20	1.48
$LOOPY_{bb}$	0.89	1.51	1.80
RCD	0.78	1.38	1.60
RCD 20×	0.60	1.09	1.31
RCD Random	0.86	1.44	1.70

[a]RMSD minima are averages over 5000 generated loops with RMSD anchor threshold 0.25 Å and benchmark set from ref 14. All RMSD values for the different and listed methods except RCD are taken from Table 2 of ref 14. Abbreviations: RCD = RCD with grid clash filter, Ramachandran pre- and inside filters, bidirectional scheme, and using the native loop, RCD 20× = RCD but sampling 20 times more (sampling number n = 100 000), RCD Random = RCD but using randomized internal coordinates instead of native ones.

dx.doi.org/10.1021/ct300977f | J. Chem. Theory Comput. 2013, 9, 1821−1829

Figure 6 shows the results of such a test using both RCD (squares, full lines) and RCD Pre (circles, dashed lines).

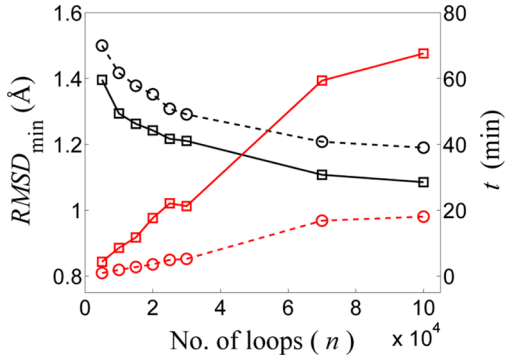


Figure 6. Evolution of averaged $\langle RMSD \rangle_{min}$ values (black) and timings (red) for undecapeptide loop cases[14] with increasing sampling number n. Circles and dashed lines depict RCD with only the Ramachandran pre-filter active (RCD Pre), whereas squares and full lines depict RCD with both the Ramachandran pre- and inside filters functioning (RCD).

Increasing sampling numbers for the undecapeptide test set were used, and similar tests were also done with octapeptides and dodecapeptides. The observations are similar for 8- and 12-peptides. The shown $\langle RMSD \rangle_{min}$ values are averages ($N = 5$), and the spread on these values was low even for small N (standard deviation $\sigma \simeq 0.04$ Å). After 18 min for the RCD Pre variant, the $\langle RMSD \rangle_{min}$ value (1.19 Å) is at the level of the observed direct tweak value (1.20 Å), which takes approximately double the time of that ($\simeq$ 36 min) in ref 14. For RCD itself with all filters active, more time is needed ($\simeq$ 67 min), but sampling is better and $\langle RMSD \rangle_{min}$ values are improved upon by approximately 0.1 Å. Both set-ups clearly show the same trend (see Figure 6) and point out that more sampling leads to conformations closer to the native loop being found. In ref 14, it was discussed that increasing the sampling number for the fastest loop closure methods does not lead to improvements in $\langle RMSD \rangle_{min}$ values unless an exceptionally high number of samplings is done (order of 1 million loop closures). For RCD, this does not appear to be the case for $\langle RMSD \rangle_{min}$ values, but we did observe that $\langle RMSD \rangle_{avg}$ values remained the same. Thus, conformational space is more screened, resulting in lower $\langle RMSD \rangle_{min}$ values, whereas the ensemble might retain its spatial characteristics. One possible explanation for the discrepancy with ref 14 might be that the discussed methods used post-filters whereas RCD uses an inside grid-clash filter.

Table 3 also gives $\langle RMSD \rangle_{min}$ values for RCD using randomized internal coordinates (RCD Random) instead of using native ones (RCD). Some evidence is present that using more realistic conditions yields somewhat higher $\langle RMSD \rangle_{min}$ values. This is certainly a relevant issue to consider when making comparisons between tests and methods in future studies.

4. CONCLUSIONS

Here, a new analytic/iterative algorithm for protein loop closure was introduced which originates in the cyclic coordinate descent approach but is distinct through new features. First, the bond selection is random, and the RMSD minimization is slightly different from previous ones. Second, conformational updating of loops is done by a hybrid spinor-matrix method, which yields a fast protocol. Third, geometric filters are integrated in the complete closure algorithm to detect clashes and steer the loop closure. Three types of geometric filters are active. A grid clash filter detects clashes with the protein environment, an interatomic distance filter eliminates loop backbone clashes, and a soft Ramachandran filter constrains the dihedral angles. Last, RCD can do loop closure in both chain directions and is steered by the convergence rates in both directions to that purpose.

Experimental results show how the different geometric filter combinations and their code placement affect sampling and time performance. Placing the grid clash and Ramachandran filters inside the loop closure routine turn out to be key points. The algorithm is accurate, fast, robust, and flexible. It outperforms CCD and other loop closure algorithms, which apply steric clash filters as post-filters. Furthermore, it is competitive with the state of the art methods in loop closure sampling. Upon increasing the sampling number, sampling performance can be further improved in RCD. As well, due to its algorithmic flexibility, it is possible to trade off sampling for time performance while retaining good overall performance. The results with RCD implicitly point out that integrating clash filters inside the loop closure routine is the way to go at present. Both the algorithms with such approach, RCD here and direct tweak as reported in ref 14, yield the best conformational loop sampling. Furthermore, RCD as a natural successor to CCD illustrates that analytic/iterative algorithms are good alternatives for protein loop closure and are competitive with the tweak algorithms.

Finally, the RCD method is very modular, so that in future work it could be integrated easily in any existing loop modeling approach. It would thus be interesting to examine the integration of RCD with knowledge-based,[12,40,41] physics-based,[10,42−45] or multiscoring strategies.[46]

■ ASSOCIATED CONTENT

Supporting Information

More detailed information about the protein loop benchmark sets used in this study. The PDF file also includes the results of the algorithm tests with RCD using the native loop protocol. This material is available free of charge via the Internet at http://pubs.acs.org/.

■ AUTHOR INFORMATION

Corresponding Author

*E-mail: denpieterch@hotmail.com (Chys), pablo@chaconlab.org (Chacón).

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

This study was supported by Spanish grants CAM-S2010/BMD-2359 and BFU2009-09552 and the Human Frontier Science Program—RGP0039/2008. We thank José Ramón López Blanco for the valuable discussions.

■ REFERENCES

(1) van den Bedem, H.; Lotan, I.; Latombe, J.; Deacon, A. *Acta Crystallogr., Sect. D* **2005**, *61*, 2−13.
(2) Go, N.; Scheraga, H. *Macromolecules* **1970**, *3*, 178−187.
(3) Shenkin, P.; Yarmush, D.; Fine, R.; Wang, H.; Levinthal, C. *Biopolymers* **1987**, *26*, 2053−2085.

(4) Chiacchio, P.; Chiaverini, S.; Sciavicco, L.; Siciliano, B. *Int. J. Robot. Res.* **1991**, *10*, 410−425.
(5) Zhao, J.; Badler, N. *ACM Trans. Graph.* **1994**, *13*, 313−336.
(6) Jamroz, M.; Kolinski, A. *BMC Struct. Biol.* **2010**, 10.
(7) Fiser, A.; Kinh Gian Do, R.; Šali, A. *Protein Sci.* **2000**, *9*, 1753−1773.
(8) Lee, J.; Lee, D.; Park, H.; Coutsias, E.; Seok, C. *Proteins* **2010**, *78*, 3428−3436.
(9) Liang, S.; Zhang, C.; Sarmiento, J.; Standley, D. *J. Chem. Theory Comput.* **2012**, *8*, 1820−1827.
(10) Lin, M.; Head-Gordon, T. *J. Chem. Theory Comput.* **2008**, *4*, 515−521.
(11) Liu, P.; Rassokhin, D.; Agrafiotis, D. *J. Comput. Chem.* **2009**, *5*, e1000478.
(12) Rata, I.; Li, Y.; Jakobsson, E. *J. Phys. Chem. B* **2010**, *114*, 1859−1869.
(13) Rossi, K.; Weigelt, C.; Nayeem, A.; Krystek, J. S. *Protein Sci.* **2007**, *16*, 1999−2012.
(14) Soto, C.; Fasnacht, M.; Zhu, J.; Forrest, L.; Honig, B. *Proteins* **2009**, *70*, 834−843.
(15) Canutesco, A. A.; Dunback, J. R. *Protein Sci.* **2003**, *12*, 963−972.
(16) Al-Nasr, K.; He, J. *Int. J. Data Mining Bioinf.* **2009**, *3*, 346−361.
(17) Boomsma, W.; Hamelryck, T. *BMC Bioinf.* **2005**, *6*, 159(11−10).
(18) Zhao, S.; Zhu, K.; Li, J.; Friesner, R. *Proteins* **2011**, *79*, 2920−2935.
(19) Cortés, J.; Siméon, T.; Remaud-Siméon, M.; Tran, V. *J. Comput. Chem.* **2004**, *25*, 956−967.
(20) Mandell, D.; Coutsias, E.; Kortemme, T. *Nat. Methods* **2009**, *6*, 552−553.
(21) Hurst, T. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 190−196.
(22) Totrov, M. In *Homology Modeling: Methods and Protocols*; Orry, A., Abagyan, R., Eds.; Methods in Molecular Biology; Springer Science: New York, 2012; Vol. *857*; Chapter Loop Simulations, pp 207−229.
(23) Jacobson, M.; Pincus, D.; Chayon, S.; Day, T.; Honig, B.; Shaw, D.; Friesner, R. *Proteins: Struct., Funct., Bioinf.* **2004**, *55*, 351−367.
(24) Ko, J.; Lee, D.; Park, H.; Coutsias, A.; Lee, J.; Seok, C. *Nucleic Acids Res.* **2011**, *39*, w210−w214.
(25) Lee, G.; Shin, W.; Park, H.; Shin, S.; Seok, C. *Bull. Korean Chem. Soc.* **2012**, *33*, 770−774.
(26) Nilmeier, J.; Hua, L.; Coutsias, E.; Jacobson, P. *J. Chem. Theory Comput.* **2010**, 1564−1574.
(27) Minary, P.; Levitt, M. *J. Comput. Biol.* **2010**, *17*, 993−1010.
(28) Xiang, Z.; Soto, C.; Honig, B. *Proc. Natl. Acad. Sci. U. S. A.* **2002**, *99*, 7432−7437.
(29) Coutsias, E.; Seok, C.; Jacobson, M.; Dill, K. *J. Comput. Chem.* **2004**, *25*, 510−528.
(30) Coutsias, E.; Seok, C.; Wester, M.; Dill, K. *Int. J. Quantum Chem.* **2006**, *106*, 176−189.
(31) Choi, V. *J. Chem. Inf. Model.* **2005**, *46*, 438−444.
(32) Chys, P.; Chacon, P. *J. Comput. Chem.* **2012**, *33*, 1717−1729.
(33) Dorst, L.; Fontijne, D.; Mann, S. *Geometric Algebra for Computer Science: An Object-oriented Approach to Geometry*; Morgan Kaufmann: San Francisco, 2007.
(34) Hestenes, D. *New Foundations for Classical Mechanics*; Reidel: Dordrecht, The Netherlands, 1986.
(35) Doran, C.; Lasenby, A. *Geometric Algebra for Physicists*; Cambridge University Press: Cambridge, U. K., 2007.
(36) Horn, B. *J. Opt. Soc. Am.* **1987**, *4*, 629−642.
(37) Fersht, A. *Enzyme Structure and Mechanism*; W.H. Freeman and Company: New York, 1985.
(38) Engh, R.; Huber, R. *Acta Crystallogr.* **1991**, *A47*, 392−400.
(39) Berkholz, D.; Shapovalov, M.; Dunbrack, R.; Karplus, P. *Structure* **2009**, *17*, 1316−1325.
(40) Rohl, R.; Strauss, C.; Chivian, D.; Baker, D. *Proteins* **2004**, *55*, 656−677.
(41) Zhang, C.; Liu, S.; Zhou, Y. *Protein Sci.* **2004**, *13*, 391−399.
(42) Felts, A.; Gallicchio, E.; Chekmarev, D.; Paris, K.; Friesner, R.; Levy, R. *J. Chem. Theory Comput.* **2008**, *4*, 855−868.
(43) de Bakker, P.; DePristo, M.; Burke, D.; Blundell, T. *Proteins: Struct., Funct., Genet.* **2002**, *51*, 21−40.
(44) Sellers, B.; Zhu, K.; Zhao, S.; Friesner, R.; Jacobson, M. *Proteins: Struct., Funct., Bioinf.* **2008**, *72*, 959−971.
(45) Arnautova, Y.; Abagyan, R.; Totrov, M. *Proteins* **2011**, *79*, 477−498.
(46) Li, Y.; Rata, I.; Jakobsson, E. *BMC Struct. Biol.* **2010**, *10*, 1−14.