A Filtering Technique for Fragment Assembly-based Proteins Loop Modeling with Constraints

F. Campeotto[1,2], A. Dal Palù[3], A. Dovier[2], F. Fioretto[1], and E. Pontelli[1]

[1] Dept. Computer Science, New Mexico State University
[2] Depts. Math. & Computer Science, University of Udine
[3] Dept. Mathematics, University of Parma

Abstract. Methods to predict the structure of a protein often rely on the knowledge of macro sub-structures and their exact or approximated relative positions in space. The parts connecting these sub-structures are called *loops* and, in general, they are characterized by a high degree of freedom. The modeling of loops is a critical problem in predicting protein conformations that are biologically realistic. This paper introduces a class of constraints that models a general multi-body system; we present a proof of NP-completeness and provide filtering techniques, inspired by inverse kinematics, that can drastically reduce the search space of potential conformations. The paper shows the application of the constraint in solving the protein loop modeling problem, based on fragments assembly.

1 Introduction

Proteins are macro-molecules of fundamental importance in the way they regulate vital functions in all biological processes. In general, there is a direct correspondence between a protein function and its 3D structure—as the structure guides the interactions among molecules. Thus, proteins structure analysis is essential for biomedical investigations, e.g., drug design and protein engineering.

The natural approach of investigating protein conformations through simulations of physical movements of atoms and molecules is, unfortunately, beyond the current computational capabilities [23, 3, 26]. This has originated a variety of alternative approaches, many based on *comparative modeling*—i.e., small structures from related protein family members are used as templates to model the global structure of the protein of interest [24, 19, 34, 28, 25]. In these methods, named *fragments assembly*, a protein structure is assembled by using small protein subunits as templates that present similarities (*homologous affinity*) w.r.t. the target sequence. The literature has also demonstrated the strength of *Constraint Programming (CP)* techniques in investigating the problem of protein structure prediction—where constraints are used to model the structural variability of a protein [1, 2, 8, 9].

In this paper, we model the problem of assembling rigid fragments as a global constraint. We abstract the problem as a general multi-body system, where each composing body is constrained by means of geometric properties and it is related to other bodies through joint relationships. This model leads to the *Joined-Multibody* (JM) constraint, whose satisfaction we prove to be NP-complete. Realistic protein models require the assembly of hundreds of different body versions, making the problem intractable. We study an efficient approximated propagator, called *JM filtering* (JMf), that allows us to efficiently compute classes of solutions, partitioned by structural similarity and controlled tolerance for error.

We implement and test the constraint and its propagator in the FIASCO framework [10]. We demonstrate our approach in addressing the *loop modeling* problem, which is a special case of the JM constraint. The goal is to connect two bodies that have fixed positions in space and that are connected by a sequence of highly variable bodies. The loop is characterized by a high degree of freedom and resembles the inverse-kinematic problem found in robotics, with some spatial constraints. We demonstrate the strength of the filtering algorithm in significantly reducing the search space and in aiding the selection of representative solutions. We compare our results, based on popular benchmark suites, to other programs specialized on loop modeling, with very encouraging results.

2 Related Work and Background

Loop modeling can be described as a special case of the protein structure prediction problem, where CP has been extensively employed. CP has been used to provide approximated solutions for *ab-initio* lattice-based modeling of protein structures, using local search and large neighboring search [33, 15]; exact resolution of the problem on lattice spaces using CP, along with with clever symmetry breaking techniques, has also been investigated [1]. These approaches solve a constraint optimization problem based on a simple energy function (HP). A more precise energy function has been used in [8, 11], where information on secondary structures (helices, sheets) are taken into consideration.

Due to the approximation errors introduced by lattice discretization, these approaches do not scale to medium-size proteins. Off-lattice models, based on the idea of fragment assembly, and implemented using Constraint Logic Programming over Finite Domains, have been presented in [9, 10] and applied not only to structure prediction but also to other structural analysis problems—e.g., the tool developed in [9] has been used to to generate sets of feasible conformations for studies of protein flexibility [13]. The use of CP to analyze NMR data and the related problem of protein docking has been studied in [2].

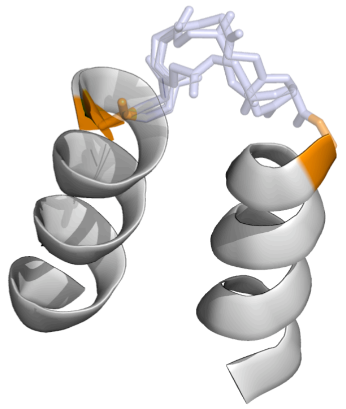


Fig. 1: Helices with a loop

Even when protein structure prediction is realized using homologous templates, the final conformation may present aperiodic structures (*loops*) connecting the known protein segments on the outer region of the protein, where the presence of the solvent lessens the restrictions on the possible movements of the structure. These protein regions are in general not conserved during evolution, and therefore templates provide very limited statistical structural information. The length of a *protein loop* is typically in the range of 2 to 20 amino acids; nevertheless, the flexibility of loops produces very large, physically consistent, conformation search spaces. Figure 1 depicts a possible scenario where two macro-structures (two helices) are connected by a loop—the loop anchors are colored in orange. The loop constraint is satisfied by the loops connecting the two anchor points. Modeling a protein loop often imposes constraints in the way of connecting two protein segments. Restrictions on the mutual positions and orientations (dihedral angles) of the loop anchors are often present. Such restrictions are defined as the *loop closure* constraints.

A procedure for protein loop modeling (e.g., [22]) typically consists of 3 phases: *sampling*, *filtering*, and *ranking*. In sampling, a set of possible loop conformations is proposed. *Ab initio* methods (e.g., [31, 17, 21, 35, 14, 16, 36]) and methods based on templates extracted from structural databases (e.g., [7]) have been explored. These conformations are checked w.r.t. the loop constraints and the geometries from the rest of the structure, and the loops that are detected as physically infeasible, e.g., causing steric clashes, are discarded by a filtering procedure. Finally, a ranking step—e.g., based on statistical potential energy (e.g., DOPE [32], DFIRE [37], or [18])—is used to select the best loop candidate(s).

Loop sampling plays an important role: it should produce structurally diverse loop conformations, in order to maximize the probability of finding one close to the native conformation. Sampling is commonly implemented as a two-step approach. First, a possible loop candidate is generated, without taking into account geometric or steric feasibility restrictions—this step usually employs dihedral angles sampled from structural databases [16]. Afterwards, the initial structure is altered into a structure that satisfies the loop closure constraints. Popular methods include the *Cyclic Coordinate Descent (CCD)* [6], the *Self-Organizing (SOS)* algorithm [30], and *Wriggling* [5]. Multi-method approaches have also been proposed—e.g., [29] proposes a loop sampling method which combines fragment assembly and analytical loop closure, based on a set of torsion angles satisfying the imposed constraints.

3 The Joined-Multibody Constraint

A *rigid block* B is composed of an ordered list of at least three (distinct) 3D points, denoted by $\mathsf{points}(B)$. The *anchors* and *end-effectors* of a rigid block B, denoted by $\mathsf{start}(B)$ and $\mathsf{end}(B)$, are the two lists containing the first three and the last three points of $\mathsf{points}(B)$. With $B(i)$ we denote the i-th point of the rigid block B. For two ordered lists of points $\boldsymbol{p}$ and $\boldsymbol{q}$, we write $\boldsymbol{p} \frown \boldsymbol{q}$ if they can be perfectly overlapped by a rigid translation/rotation (i.e., a *roto-translation*).

Definition 1 (Multi-body). *A* multi-body *is a sequence $S_1, \ldots, S_n$ of non-empty sets of rigid blocks. A sequence of rigid blocks $B_1, \ldots, B_n$, is called a* rigid body *if, for all $i = 1, \ldots, n-1$, $\mathsf{end}(B_i) \frown \mathsf{start}(B_{i+1})$. A* compatible multi-body *is a multi-body where for all pairs of rigid blocks $B, B' \in \bigcup_{i=1}^{n} S_i$ and for all $\boldsymbol{p}, \boldsymbol{q} \in \{\mathsf{start}(B), \mathsf{start}(B'), \mathsf{end}(B), \mathsf{end}(B')\}$ it holds that $\boldsymbol{p} \frown \boldsymbol{q}$.*

A rigid body can be seen as one instance of a multi-body that guarantees the partial overlapping of each two consecutive blocks. The overlapped points $\mathsf{end}(B_i)$ and $\mathsf{start}(B_{i+1})$ constitute the i-th *joint* of the rigid body. The number of rigid bodies "encoded" by a single multi-body is bounded by $\Pi_{i=1}^{n} |S_i|$.

Figure 2 provides a schematic representation of a rigid body. The joints connecting two adjacent rigid blocks are marked by orange rectangles and grey circles. The points in $\mathsf{points}(B)$ of each rigid block are represented by circles. Each rigid block extends from the first point of a joint to the last point of the successive joint.

A rigid body is defined by the overlap of joints, and relies on a chain of relative roto-translations of its blocks. Each $\mathsf{points}(B_i)$ is therefore positioned according to the (homogeneous) coordinate system associated to a rigid block B_{i-1}. Note that once the reference system for B_1 is defined, the whole rigid

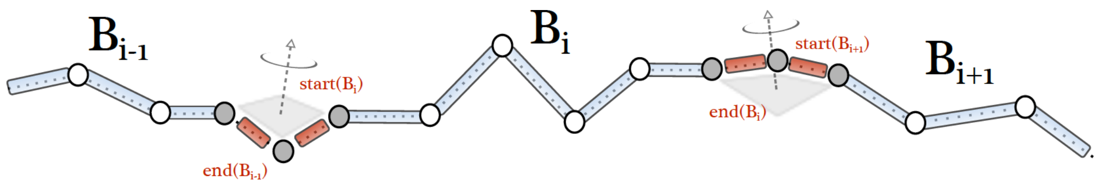


Fig. 2: A schematic representation of a rigid body

body is completely positioned.[4] The relative positions of two consecutive rigid blocks B_{i-1} and B_i of a rigid body ($2 \leq i \leq n$) can be defined by a transformation matrix $T_i \in \mathbb{R}^{4\times 4}$. Each matrix depends on the standard Denavit-Hartenberg parameters [20] obtained from the start and end of the blocks (c.f., [27] for details). We denote the product $T_1 \cdot T_2 \cdot \ldots \cdot T_i \cdot (x, y, z, 1)^T$ by $\nabla_i(x, y, z)$.

Let us focus on the matrix T_1. The block B_1 can be rigidly moved in a desired position and orientation on the basis of additional spatial constraints (e.g., the sets $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$ in the Def. 2). T_1 is a matrix that allows a roto-translation of B_1 in a position fulfilling the additional constraints.

For $i = 1, \ldots, n$, the coordinate system conversion (x', y', z'), for a point $(x, y, z) \in \mathsf{points}(B_i)$ into the coordinate system of B_1, is obtained by:

$$(x', y', z', 1)^T = T_1 \cdot T_2 \cdot \ldots \cdot T_i \cdot (x, y, z, 1)^T = \nabla_i(x, y, z) \tag{1}$$

Homogeneous transformations are such that the last value of a tuple is always 1. Note that a modification of the matrix T_1 is a sufficient step to place the whole rigid body into a different start position.

Definition 2 (JM-constraint). *The* joined-multibody (JM) constraint *is described by a tuple:* $J = \langle \boldsymbol{S}, \boldsymbol{V}, \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{E}}, \delta \rangle$, *where:*

- $\boldsymbol{S} = S_1, \ldots, S_n$ *is a multi-body. Let* $\mathcal{B} = \{B_1, \ldots, B_k\}$ *be the set of all rigid blocks in* $\boldsymbol{S}$, *i.e.,* $\mathcal{B} = \bigcup_{i=1}^n S_i$.
- $\boldsymbol{V} = V_1, \ldots, V_n$ *is a list of finite-domain variables. For* $i = 1, \ldots, n$, *the variable* V_i *is associated to a domain* $\mathsf{dom}(V_i) = \{j : B_j \in S_i\}$.
- $\boldsymbol{\mathcal{A}} = \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$, *and* $\boldsymbol{\mathcal{E}} = \mathcal{E}_1, \ldots, \mathcal{E}_{3n}$ *are lists of sets of 3D points such that:*
  - $\mathcal{A}_1 \times \mathcal{A}_2 \times \mathcal{A}_3$ *is the set of admissible points for* $\mathsf{start}(B)$, *with* $B \in S_1$;
  - $\mathcal{E}_{3i-2} \times \mathcal{E}_{3i-1} \times \mathcal{E}_{3i}$ *is the set of admissible points for* $\mathsf{end}(B)$, *with* $B \in S_i$, $i = 1, \ldots, n$;
- δ *is a constant, used to express a minimal distance constraint between different points. Let us assume that for all* $B \in \mathcal{B}$ *and for all* $a, b \in \mathsf{points}(B)$, *if* $a \neq b$ *then* $\|a - b\| \geq \delta$ *(where* $\|\cdot\|$ *is the Euclidean norm).*

A solution *for the JM constraint* J *is an assignment* $\sigma : \boldsymbol{V} \longrightarrow \{1, \ldots, |\mathcal{B}|\}$ *s.t. there exist matrixes* $T_1, \ldots, T_n$ *(used in* ∇*) with the following properties:*

<u>*Domain:*</u> *For all* $i = 1, \ldots, n$, $\sigma(V_i) \in \mathsf{dom}(V_i)$.

<u>*Joint:*</u> *For all* $i = 1, \ldots, n-1$, *let* $(a^1, a^2, a^3) = \mathsf{end}(B_{\sigma(V_i)})$ *and* $(b^1, b^2, b^3) = \mathsf{start}(B_{\sigma(V_{i+1})})$, *then it holds that (for* $j = 1, 2, 3$*):*

$$\nabla_i(a_x^j, a_y^j, a_z^j) = \nabla_{i+1}(b_x^j, b_y^j, b_z^j)$$

<u>*Spatial Domain:*</u> *Let* $(a^1, a^2, a^3) = \mathsf{start}(B_{\sigma(V_1)})$, *then* $T_1 \cdot a^j \in \mathcal{A}_j \times \{1\}$.

For all $i = 1, \ldots, n$, *let* $(e^1, e^2, e^3) = \mathsf{end}(B_{\sigma(V_i)})$ *then*

$$\nabla_i(e_x^j, e_y^j, e_z^j) \in \mathcal{E}_{3(i-1)+j} \times \{1\}$$

where $1 \leq j \leq 3$ *and* $T_2, \ldots, T_i$ *(in* ∇_i*) are the matrices that overlap* $B_{\sigma(V_{i-1})}$ *and* $B_{\sigma(V_i)}$ *(the product* $\times\{1\}$ *is due since we use homogeneous coordinates).*

<u>*Minimal Distance:*</u> *For all* $j, \ell = 1, \ldots, n$, $j < \ell$, *and for all points* $a \in \mathsf{points}(B_{\sigma(V_j)})$ *and* $b \in \mathsf{points}(B_{\sigma(V_\ell)})$, *it holds that:*

$$\|\nabla_j(a_x, a_y, a_z) - \nabla_\ell(b_x, b_y, b_z)\| \geq \delta$$

A JM constraint is said to be compatible *if* $S_1, \ldots, S_n$ *is a compatible multi-body.*

If there are no joints with the three points aligned, $T_2, \ldots, T_n$ depend deterministically from T_1 and σ. Compatible JM constraints are interesting for our target application. Nevertheless, the additional restriction does not simplify constraint solving, as we discuss below.

Complexity Analysis. The problem of determining *consistency* of JM constraints (i.e., the existence of a solution) is NP-complete. To prove this fact, we start from the NP-completeness of the consistency problem of the constraint *Self-Avoiding-Walk* (*SAW-constraint*) in a discrete lattice, proved in [12]. In particular, we will use the 3D cubic lattice for this problem. Let $\boldsymbol{X} = X_1, \ldots, X_n$ be a list of variables. Each variable has a finite domain $\mathsf{dom}(X_i) \subseteq \mathbb{Z}^3$. $\sigma : \boldsymbol{X} \longrightarrow \mathbb{Z}^3$ is a solution of the SAW constraint if:

[4] With the exception of the case where the all points of a joint are collinear.

- For all $i = 1, \dots, n$: $\sigma(X_i) \in \mathsf{dom}(X_i)$,
- For all $i = 1, \dots, n-1$: $\|\sigma(X_i) - \sigma(X_{i+1})\| = 1$,
- For all $i, j = 1, \dots, n$, $i < j$, it holds that $\|\sigma(X_i) - \sigma(X_j)\| \geq 1$.

As emerges from the proof in [12], the problem of determine the consistency of a SAW constraint is NP complete even if the domains of $\mathsf{dom}(X_1)$ and $\mathsf{dom}(X_2)$ are singleton sets. Without loss of generality, we can concentrate on SAW problems where $\mathsf{dom}(X_1) = \{(0,0,0)\}$ and $\mathsf{dom}(X_2) = \{(0,1,0)\}$—the other cases can be reduced to this one using a roto-translation.

Theorem 1. *The consistency problem for the JM constraint is NP-complete.*

Proof (sketch). The proof of membership in NP is trivial; given a tentative solution, it is easy to test it in polynomial time—the most complex task is building the rotation matrixes.

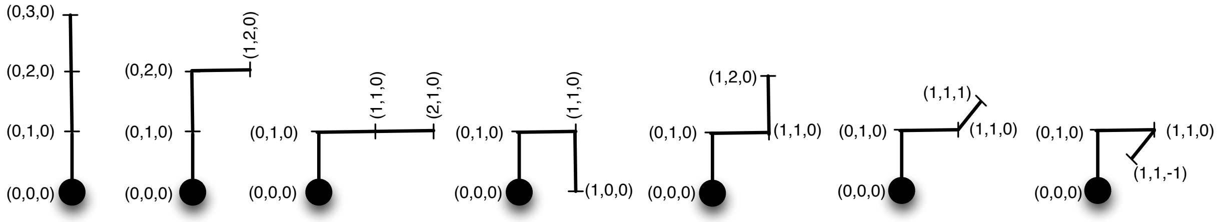


Fig. 3: Rigid blocks for SAW (from left to right, block 1, . . . , 7)

To prove completeness, let us reduce SAW, with the further hypothesis on $\mathsf{dom}(X_1)$ and $\mathsf{dom}(X_2)$ to JM. Let us consider an instance $A = \langle \boldsymbol{X}, \mathsf{dom}(\boldsymbol{X})\rangle$ of SAW with n ($n > 3$) variables. We define a equi-satisfiable instance $B = \langle \boldsymbol{S}, \boldsymbol{V}, \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{E}}, \delta\rangle$ of JM as follows. Let us choose $\boldsymbol{V} = V_1, \dots, V_{n-3}$. We select the sets S_i of rigid blocks of the multi-body to be all identical, and consisting of all the (non overlapping) fragments of three contiguous unitary segments of length 1, starting from $(0,0,0)$ and with bends of 0 or 90 degrees. A filtering using symmetries is made and the blocks are indicated in Fig. 3. For all $i = 1, \dots, n-3$ we assign the following sets of 3D points to the end-effectors:

$$\mathcal{E}_{3i-2} = \mathsf{dom}(X_{i+1}) \cap \mathbb{Z}^3, \quad \mathcal{E}_{3i-1} = \mathsf{dom}(X_{i+2}) \cap \mathbb{Z}^3, \quad \mathcal{E}_{3i} = \mathsf{dom}(X_{i+3}) \cap \mathbb{Z}^3$$

Moreover, let $\mathcal{A}_1 = \{(0,0,0)\}, \mathcal{A}_2 = \{(0,1,0)\}, \mathcal{A}_3 = \{(0,2,0),(1,1,0)\}$. Observe that all these sets are subsets of $\mathbb{Z}^3$ and therefore points of the same lattice of the SAW problem. Assigning $\delta = 1$, the reduction is complete. It is immediate to check that SAWs in the 3D lattice and the solutions of the JM constraint defined via reduction are essentially the same 3D polygonal chain. $\square$

We have a proof for the NP-completeness of the compatible JM constraint (in particular, it does not make use of joints made by collinear points), by reduction from the Hamiltonian path problem in special planar graphs. The proof is omitted due to lack of space. The interested reader can find it at `http://www.cs.nmsu.edu/fiasco`.

4 Filtering algorithm for the joined-multibody constraint

Since checking the satisfiability (and hyper-arc consistency) of the JM constraint is NP-complete, we studied an approximated polynomial time filtering algorithm. When dealing with multi-bodies, the computation of the end-effectors' spatial domains provides limited filtering information, since it identifies a large volume.

We designed an algorithm (JMf, Algorithm 1) that is inspired by bound-consistency on the 3D positions of end-effectors. The algorithm uses an equivalence (clustering) relation over these bounds, in order to retain precise information about classes of domain variable assignments that produce similar spatial results. This allows a compact handling of the combinatorics of the multi-body, while a controlled error threshold allows us to select the precision of the filtering. The equivalence relation captures those rigid bodies that are geometrically similar and thus compacts small differences among them; relevant gains in computation time can be derived when some errors are tolerated.

The JMf algorithm receives as input a JM-constraint $\langle \boldsymbol{S}, \boldsymbol{V}, \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{E}}, \delta\rangle$, along with a set $\mathcal{G}$ of points that are not available for the placement of bodies (e.g., points of other parts of the protein we are studying, points of the loop non-deterministically assigned by previous calls of the algorithm itself) and an equivalence relation $\sim$ on the space of triples of 3D points. The algorithm makes use of a function M (line 8); this function takes as input two lists $\boldsymbol{a}$ and $\boldsymbol{b}$ of 3D points, and computes the homogeneous transformation to overlap $\boldsymbol{b}$ on $\boldsymbol{a}$. A call to this function will fail if $\boldsymbol{a} \not\sim \boldsymbol{b}$. For simplicity, the fourth component (always 1) of the homogeneous transformation is not explicitly reported in the algorithm.

Algorithm 1 The JMf algorithm.

```
Require: $\boldsymbol{S}, \boldsymbol{V}, \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{E}}, \mathcal{G}, \delta, \sim$
 1: $n \leftarrow |\boldsymbol{V}|$
 2: $\mathcal{R}_1 \leftarrow \left\{ B \in S_1 \,\middle|\, \exists T_1 \begin{pmatrix} T_1 \cdot \mathsf{start}(B) \in \mathcal{A}_1 \times \mathcal{A}_2 \times \mathcal{A}_3 \;\wedge \\ T_1 \cdot \mathsf{end}(B) \in \mathcal{E}_1 \times \mathcal{E}_2 \times \mathcal{E}_3 \quad\; \wedge \\ \forall p \in \mathsf{points}(B). \forall q \in \mathcal{G}. \, \|(T_1 \cdot p) - q\| \geq \delta) \end{pmatrix} \right\}$
 3: $\mathcal{P}_1 \leftarrow \{T_1 \cdot \mathsf{end}(B) \;\mid\; B \in \mathcal{R}_1, T_1$ as in the line above $\}$
 4: $\mathsf{dom}(V_1) \leftarrow \{\mathsf{label}(B) \;\mid\; B \in \mathcal{R}_1\}$
 5: for all $i = 2, \dots, n$ do
 6:     $\mathcal{P}_i = \emptyset$; $\mathcal{R}_i = \emptyset$;
 7:     for all $E \in \mathcal{P}_{i-1}$ do
 8:         $\mathcal{R}_i \leftarrow \mathcal{R}_i \cup \left\{ B \in S_i \,\middle|\, \begin{array}{l} T = M(E, \mathsf{start}(B)) \;\wedge\; T \neq \mathtt{fail} \;\wedge \\ T \cdot \mathsf{end}(B) \in \mathcal{E}_{3i-2} \times \mathcal{E}_{3i-1} \times \mathcal{E}_{3i} \quad \wedge \\ \forall p \in \mathsf{points}(B). \forall q \in \mathcal{G}. \, \|(T \cdot p) - q\| \geq \delta \end{array} \right\}$
 9:         $\mathcal{P}_i \leftarrow \{M(E, \mathsf{start}(B)) \cdot \mathsf{end}(B) \;\mid\; B \in \mathcal{R}_i\}$
10:     end for
11:     compute $\mathcal{P}_i/\!\sim$ and filter $\mathcal{R}_i$ accordingly
12:     $\mathsf{dom}(V_i) \leftarrow \{\mathsf{label}(B) \;\mid\; B \in \mathcal{R}_i\}$
13: end for
```

For $i = 1, \dots, n$, the algorithm computes the sets $\mathcal{R}_i$ and $\mathcal{P}_i$, that will respectively contain the blocks from S_i that can still lead to a solution, and the corresponding allowed 3D positions of their end-effectors. These two sets are strongly related: a data structure linking each block $B \in S_i$ with the list of its corresponding possible end-effectors (and vice versa) is used at the implementation level. For each block $B \in \mathcal{B}$, we denote with $\mathsf{label}(B)$ a unique label identifying it; $\mathsf{dom}(V_i)$ is therefore the set of the blocks' labels that can be used for the variable V_i. The sets $\mathcal{R}_i$ and $\mathcal{P}_i$ are used to determine the domain $\mathsf{dom}(V_i)$ of the variable V_i (lines 4 and 12). In computing/updating $\mathcal{R}_i$ and $\mathcal{P}_i$, only blocks that have end-effectors contained in the bounds $\mathcal{E}_{3i-2}, \mathcal{E}_{3i-1}, \mathcal{E}_{3i}$ are kept. Fragments that would cause points to collapse—i.e., due to a distance smaller than δ from previously placed points—are filtered out (lines 2 and 8). Moreover, the spatial positions of the points of the first block are validated against $\mathcal{A}$ (line 2).

The algorithm performs $|\boldsymbol{V}| - 1$ iterations (lines 5–13). First $\mathcal{R}_i$ and $\mathcal{P}_i$ are computed on the basis of the sets of end-effectors of the previous level $\mathcal{P}_{i-1}$ and the starting point of a selected block B, filtering out those that are not overlapping and those that lead to wrong portions of space (lines 8–9). Then, the $\sim$-based clustering filtering is applied (line 11). During this step, the set of triples of 3D points $\mathcal{P}_i$ is clustered using $\sim$. A representative of each equivalence class is chosen (within $\mathcal{P}_i$). The corresponding block in $\mathcal{R}_i$ is marked. All non marked blocks are filtered out from $\mathcal{R}_i$. Let us also note that the filtering based on clustering is not performed for the initial step $\mathcal{P}_1$, as typically this is already captured by the restrictions imposed by $\mathcal{A}$.

In our tests, the initial domains $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ are singleton sets (we start from a rigid block with known end-effectors). Moreover, w.l.o.g., all initial fragments matching with those three points are rotated in the same reference in our database, allowing to use a unique T_1 for all blocks (lines 2 and 3). We experimentally verified that avoiding clustering in the first stage allows to sensibly reduce approximation errors. The filtering algorithm is similar to a directional arc-consistency, where the global constraint is viewed as a conjunction of binary constraints between adjacent blocks. Its peculiarity, however, is the use of representative clustered triples to compactly store the domains.

5 Loop modeling by the joined-multibody constraint

In this section, we use the joined-multibody constraint to model the protein loop problem. We have implemented the proposed encoding and the constraint solving procedure, based on filtering Algorithm 1, within *FIASCO* (Fragment-based Interactive Assembly for protein Structure prediction with COnstraints) [4].

FIASCO is a C++ tool that provides a flexible environment that allows us to easily manipulate constraints targeted at protein modeling. These constraints do not only concern geometric and energetic aspects but, in general, they allow one to model particular portions of the target protein using arbitrarily long homolog structures. A protein is modeled through the sequence of its amino acids. The *backbone* of each amino acid is represented by all its atoms: $N, C\alpha, C', O$. A single point CG is used for representing the *side chain*, namely the set of atoms characteristic of each amino acid (Fig. 4). *fragment assembly* is used to restrict the allowed spatial positions of consecutive backbone atoms. The final protein conformation is built by combining fragments, treated as basic assembly units. A fragment with $h \geq 1$ amino acids is the concatenation of $4h + 3$ atoms, represented by the regular expression $C' \; O \; (N \; C\alpha \; C' \; O)^h \; N$. The assembling of two fragments is performed by overlapping the planes β_R and β_L, determined by the atoms C', O, N ending the first fragment and starting the second fragment (Fig. 4). The overlapping is made

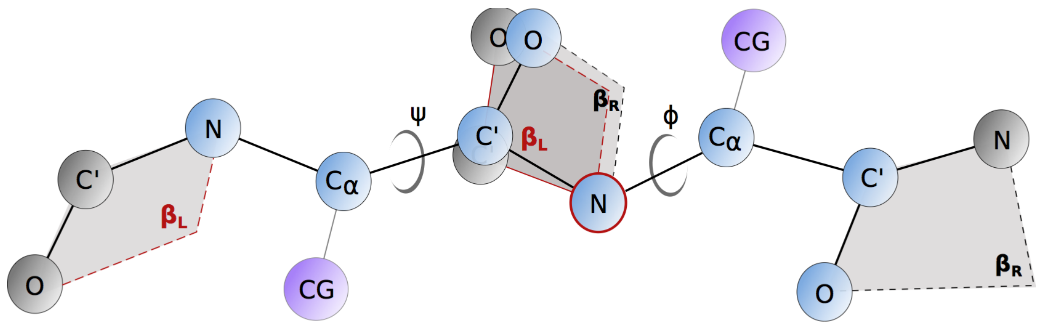


Fig. 4: Fragments are assembled by overlapping the plane β_R, described by the rightmost C', O, N atoms of the first fragment (left), with the plane β_L, described by the leftmost C', O, N atoms of the second fragment (right), on the common nitrogen atom

in the N atom in order to best preserve the torsion angle ϕ characteristic of the first amino acid of the second fragment. Each backbone atom is represented by a `Point` variable, whose initial domain can be described by a 3D interval $[\boldsymbol{L}, \boldsymbol{U}]$, that identifies the lower and the upper bounds of a 3D box enclosing the set of possible positions for the atom.[5] A second set of variables (`Fragment`) is used to maintain the sets of possible fragments that can be used to model different segments of the protein. The respective domains are the sets of elements that link the specific protein region modeled by a `Fragment` variable to the related `Point` variables for the atoms in such region. Constraints are introduced to link the domains of `Fragment` variables with those of related `Point` variables.

Loop Modeling. Let us now build on the core constraints of FIASCO and on the JM constraint to address the loop modeling problem. The starting point is a given protein with two known (large) blocks. The model will account for them in the definitions of sets $\boldsymbol{\mathcal{E}}$ (they also allow us to build the set $\mathcal{G}$—see Algorithm 1). The start of the first block and the end of the last block, namely a sequence $C'ON$ (initial anchor) of coordinates $\boldsymbol{a} = (a^1, a^2, a^3)$, and a sequence $C'ON$ (final anchor) of coordinates $\boldsymbol{e} = (e^1, e^2, e^3)$ are known, as well as the sequence $x_1, \ldots, x_n$ of amino acids connecting these two points. Each amino acid can have different 3D forms, depending on the angles Ψ, Φ and the position of CG; a statistically pre-computed set of these forms is loaded from a repository (e.g. `www.pdb.org`) and a weight depending on its frequency is assigned to each fragment. Each fragment is identified by an integer label. Loop modeling can be realized using the joined-multibody constraint $J = \langle \boldsymbol{S}, \boldsymbol{V}, \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{E}}, \delta \rangle$ where:

- For $i = 1, \ldots, n$ the set S_i contains all the fragments associated with the amino acid x_i.
- For $i = 1, \ldots, n$, $\mathsf{dom}(V_i)$ is the set of the labels of the fragments in S_i.
- The constant δ (now $\delta = 1.5\text{Å}$) asserts a minimum distance between atoms.
- For the spatial domains, for $j = 1, \ldots, 3$, we set $\mathcal{A}_j = \{a^j\}$ and $\mathcal{E}_{3(n-1)+j}$ is the 3D interval $[e^j - (d_j, d_j, d_j), e^j + (d_j, d_j, d_j)]$, where the values d_j are derived from the covalent radii bond distances $\epsilon_N, \epsilon_O, \epsilon_C$ of the specific types of atoms (specifically, $d_1 = \epsilon_N, d_2 = d_1 + \epsilon_O, d_3 = d_2 + \epsilon_C$). We use this slack for the last 3 points of the loop in order to cushion the error produced during the clustering step, still obtaining solutions that are geometrically eligible.
  For $\mathcal{E}_1, \ldots, \mathcal{E}_{3(n-1)}$ we allow a "sufficiently large" box of spatial points. Precisely, each $\mathcal{E}_i$ is the box obtained by enlarging by $4n$Å in all the directions, the box identified by the two points $[\boldsymbol{a}, \boldsymbol{e}]$ (or $[\boldsymbol{e}, \boldsymbol{a}]$) — 4Å is a rough upper bound to the distance between two consecutive C' in a protein.[6]

Let us observe that the fragments considered lead to a *compatible* multi-body (Def. 1)—thanks to the use of a full-atom description of fragments. A different level of description of the fragments (e.g., the $C\alpha$–$C\alpha$ modeling used in [9] for the fragment-assembly approach to the complete protein folding) would not lead us to a compatible multi-body. Moreover, known larger rigid blocks can be easily inserted in the modeling in an explicit way in some S_i. More loops can be also modeled simultaneously in this way.

Clustering. The JMf algorithm is parametric w.r.t. the clustering relation and the function selecting the representative; they both express the degree of approximation of the rigid bodies to be built. The proposed clustering relation for loop modeling takes into account two factors: *(a)* The positions of the end-effectors in the 3D space and *(b)* The orientation of the planes formed by the fragment's anchor β_L and end-effector β_R (Fig. 4). This combination of clusterings allows to capture local geometrical similarities, since both spatial and rotational features are taken into account.

[5] In the current implementation of FIASCO initial domains can be just boxes.

[6] $\mathcal{E}_i$ should be intersected with the complement of $\mathcal{G}$ (the region of space occupied by the two known rigid blocks). This kind of domain operation is not yet supported by FIASCO and therefore the control of this part is handled by Algorithm 1.

The spatial clustering *(a)* used is the following. Given a set of fragments, the three end points $C'ON$ (end effectors) of each cluster are considered, and the centroid of the triangle $C'ON$ is computed. We use three parameters: $k_{min}, k_{max} \in \mathbb{N}$, $k_{min} \leq k_{max}$, and $r \in \mathbb{R}$, $r \geq 0$. We start by selecting a set of k_{min} fragments, pairwise distant at least $2r$. These fragments are selected as representatives of an equivalence class for other fragments that fall within a sphere of radius r centered in the centroid of the representative. This clustering ensures a rather even initial distribution of clusters, however some fragments may not fall within the k_{min} clusters. We allow to create up to $k_{max} - k_{min}$ new clusters, each of them covering a sphere of radius r. Remaining fragments are then assigned to the closest cluster. Other techniques can be employed, the one used allows a fast implementation and acceptable results.

The orientation clustering *(b)* partitions the fragments according to their relative orientation of planes β_R, β_L (and it can be pre-computed, being independent on the roto-translation the fragment will be subject to). Let us observe that here we consider all four points of a fragment. All fragments are characterized by the normal to the plane β_R, assuming that each fragment is already joined to the previous one. The clustering algorithm guarantees that each cluster contains fragments that pair-wise have normals with a deviation of at most a threshold β degrees. This algorithm produces a variable number of partitions depending on β.

The final cluster is the intersection of the two partitioning algorithms. This defines an equivalence relation $\sim$ depending on k_{min}, k_{max}, r, and β. The representative selection function selects the fragment for each partition according to some preferences (e.g., most frequent fragment, closest to the center, etc.).

Note that for $r = 0$, $\beta = 0$, and k_{max} unbounded, no clustering is performed and this would cause the combinatorial explosion of every possible end-effector on the whole problem. The spatial error introduced depends on r and β. With $\beta = 0$, the error introduced at each step can be bound by $2r$ for each dimension. At each iteration the errors are linearly increased, since a new fragment is placed with an initial error gathered from previous iterations, thus resulting in a $2nr$ bound for the last end-effector. Clearly this bound is very coarse, and on average the experimental results show better performances. Similar considerations can be argued for rotational errors, however the intersection of the two clusterings, provide, in general, a much tighter bound.

Implementation Details. A pre-processing step clusters "a-priori" the variables' domains (by means of a first call of the JMf algorithm with $\mathcal{G} = \emptyset$). The rotational clustering is made while the fragment databases is computed. This speeds-up the clustering algorithm during the search, since at that time the spatial clusters can only be intersected with the ones already computed.

As soon as the domain for the variables related to the initial anchor of a JM constraint is instantiated, the corresponding constraint is woken up. The algorithm JMf is invoked with the parameters described above. If there are no empty domains after this stage, the search proceeds by selecting the leftmost variable and assigning it a fragment (block) in a leftmost order. All domains are pre-sorted from the most likely to the least likely for each variable (the previous stage of filtering preserves the ordering).

6 Experimental Results

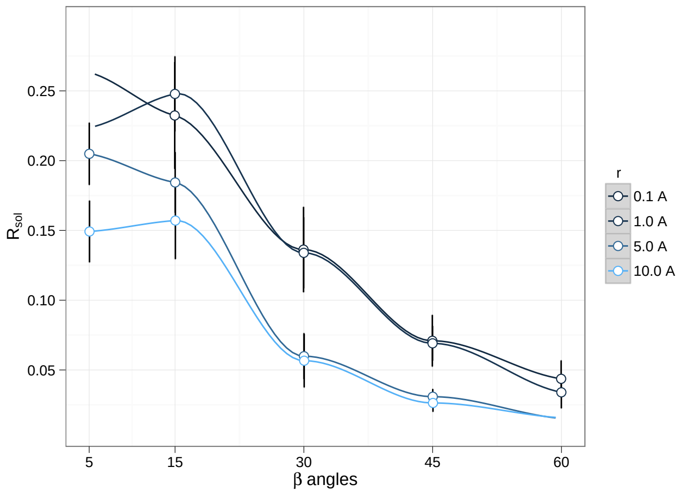
We report on the experimental results obtained from a prototype implementation of the JM constraint, along with its Joined-Multibody filtering algorithm, in the FIASCO system. The system, protein lists, and some examples are available at `http://www.cs.nmsu.edu/fiasco`. Experiments are conducted on a Linux Intel Core i7 860, 2.5 GHz, memory 8 GB, machine. The proposed method has been applied to a data set of 10 loop targets for each of the lengths 4, 8, and 12 residues. The targets are chosen from a set of non-redundant X-ray crystallography structures [6].

We first analyze the performances of JMf filtering by examining the fraction of the search space explored during solution search. Next, we compare the qualities of the loop conformations generated, by measuring the root mean square deviation (RMSD) of the proposed loop with respect to the native conformation. RMSD captures the overall similarity in space of corresponding atoms.

6.1 Filtered search space and performances

For each of the 10 selected proteins with a loop of length $n = 4$ we compare two CSPs. The first one enables the JM constraint and the second one disables it (simple combinatorial fragment assembly). For both problems we have exhaustively generated (without timeout) all the solutions by means of a constraint-based search using a classical propagate-labeling schema

The number of fragments in each variable domain is 60—this is an adequate sampling to describe a reasonable amino-acid flexibility. This increases the likelihood to generate a loop structure that is similar to the native one. A loop of length n generates an exponential search space of size roughly 60^n ($\sim 1.3 \cdot 10^7$ in this example). The selected variable is the leftmost one. Fragments are selected in descending frequency



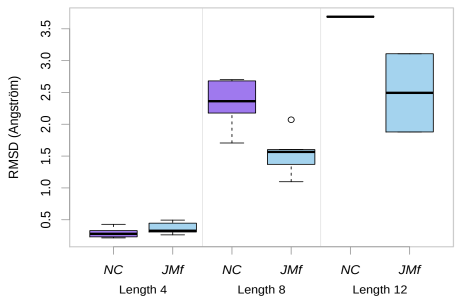


Fig. 5: Ratio of the solutions (left) and RMSD comparison (right)

order. We have used different values for r and β, while k_{min} and k_{max} are set to 20 and 100, respectively. In Fig. 5 (left) we show, for different combinations of clustering parameters, the ratio ($\mathsf{R_{sol}}$) between the number of solutions to the CSPs with JM constraint and without it. The size of the prop-labeling tree and running times decrease with a similar trend.

The adopted parameters for the β angles are reported in the x-axes, while the r values for the clustering distances are plotted in different colors (10Å is the lightest color). The white dots represent the average values of all the trials and the vertical bars illustrate the standard error of the mean: $\frac{\sigma}{\sqrt{N}}$, where σ is the standard deviation and N is the number of samples. It can be observed that the number of the solutions generated by the JM constraint decreases as the β and r values increase, as one can expect.

When the clustering is too inaccurate, for some loop targets no solutions are found by the JM constraint (e.g., with $\beta = 60$ and $r = 10.0$ for 6 loops out of 10). In Section 6.2 we show that the reduced number of solutions does not affect the approximate completeness of the search.

Let us analyze the case with longer loops (10 proteins of lengths 8 and 10 of lengths 12) where the set of all possible solutions without using JMf cannot be computed in reasonable time. In order to estimate the filtering capability, we build an approximation based on the following algorithm. For $i = 1, \ldots, n$:

- Let us assume we have non-deterministically assigned the variable V_i.
- The variables $V_{i+1}, \ldots, V_n$ have the domains $\mathsf{dom}(V_j) = d_j$, for $j = i+1, \ldots, n$.
- We apply the JM filtering algorithm obtaining the new values $d'_{i+1}, \ldots, d'_n$
- An approximation to the portion of the search space removed at this stage is computed as: $\prod_{j=i+1}^{n} d_j - \prod_{j=i+1}^{n} d'_j \quad (2)$.

To compute the search space ξ removed by the propagation of the JM constraint we sum the values (2) for every node of the search tree visited within the timeout. Let $\varrho = 1 + \sum_{i=1}^{n} \prod_{j=i}^{n} \mathsf{dom}(V_j)$ be the size of the search tree in absence of constraint propagation. In Figure 6 we report the behavior of $\frac{\varrho - \xi}{\varrho}$ for targets loops of length 8 and 12, depending on r and β. Let us observe that ϱ is a rough upper bound of the nodes that really need to be visited (several fragments can immediately lead to a failure due to spatial constraints) and that allotting more time for the computations allows further pruning, thus increasing ξ.

We have set two timeouts: one involving solely the exploration of the prop-labeling tree, fixed to 600 seconds, and another over the total computation (search tree exploration and JM propagation), fixed to 2 hours. In every trial carried out we abort the search due to timeout in exploring the prop-labeling tree. The space filtered by the JM constraint rises according to the increasing values of r and β. In the settings proposed, fixing the values of r and β to 10.0 and 60 respectively, the propagation of the JM constraint produces a filtering over the search space that allows the rest of the search to be carried out merely on about the 10 % of the original prop-labeling tree. However, as stated above, this is just an under-estimation of the pruning capability.

6.2 Loop Conformations quality

This section provides some insight about the quality of the solutions that are retrieved when using the JM constraint. In particular, we show that the quality in terms of RMSD is not significantly degraded by the large filtering performed by the propagator. As in the previous subsection, we use as reference for the comparisons the results from the search with no JM constraint (named NC). The experiments were carried on with $k_{min} = 20$ and $k_{max} = 100$, while r and β are set respectively to 1.0 and 15 for loops of

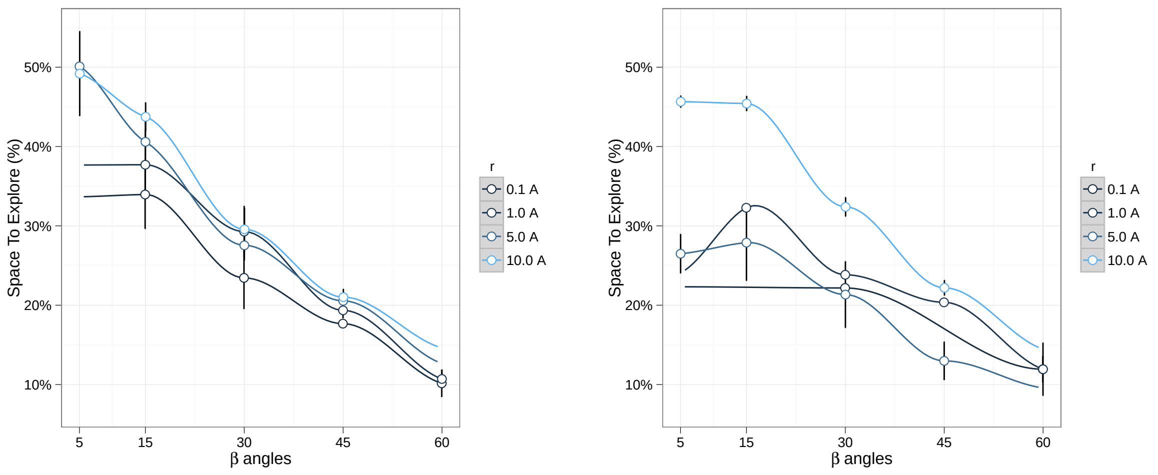


Fig. 6: Ratio of the search space explored using JMf for 8-loops (left) and 12-loops.

length 4, and 2.5 and 60 for loops of length 8 and 12. In our analysis, such parameters guarantee a good compromise between filtering power and accuracy of the results.

In Figure 5 (right), the bottom and top point of each vertical line show the RMSD of the best and worst prediction, respectively, within the group of targets analyzed. The results are biased by the fragment database in use: we excluded from it the fragments that belong to the deposited protein targets. Therefore it is not possible to reconstruct the original target loop and none of the searched are expected to reach a RMSD equal to 0. The bottom and top horizontal lines on each box shows the RMSD of the 25^{th} and 75^{th} percentile prediction, respectively, while the line through the middle shows the median. We observe no substantial difference in the distributions related to short loop predictions (length 4), and an improvement for targets of greater size due to time-out. Such results experimentally show the strength of our method: JM filtering algorithm removes successfully redundant conformations; moreover, it quickly direct the search space exploration through predictions that are biologically meaningful.

In Figure 7, we analyze the impact of the k_{max} on computational times (left) and precision (right) of the filtering of the JM constraint. The tests are performed over the protein loops of length 4 adopting as cluster parameters, $r = 1.0$, $\beta = 30$, $k_{min} = 20$. We ignore minimum and maximum values from each data set to smooth out fluctuations and highlight average trends. Each dot in the plot represents the outcome of a trial and the grey area denote the standard error of the mean associated to a particular value of k_{max}. The RMSD values tend to decrease as the number of clusters increase, and it stabilizes when $k_{max} \geq 500$ clusters with a good average value of 0.4Å. As one might expect, instead, the filtering time increases as k_{max} increases.

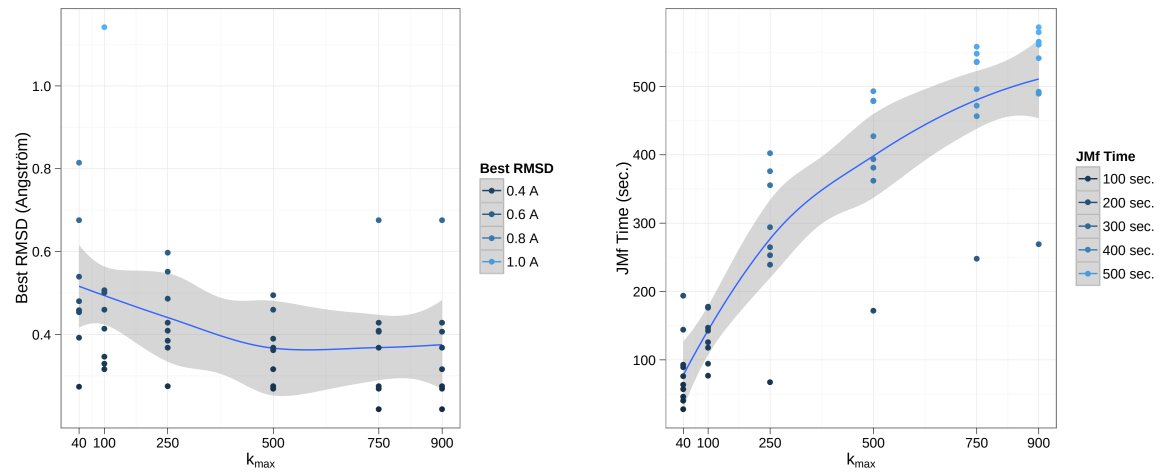


Fig. 7: Comparison of the best RMSD values (left) and JMf computation times (right) at varying of the k_{max} clustering parameter.

6.3 Comparison with other state-of-the-art loop samplers

We also compare our method to three other state-of-the-art loop samplers: the Cyclic Coordinate Descent (CCD) algorithm [6], the self-organizing algorithm (SOS) [30], and the FALCm method [29]. Note that our solution does not include specific heuristics and additional information that are used in the other programs. Moreover, our database is not tuned for loop prediction: it is built from any fragment that may appear on a protein (e.g. including helices, β-strands).

Table 1 shows the average RMSD for the benchmarks of length 4, 8 and 12 as computed by the four programs. We report the results as given in Table 2 of [6] for the CCD, Table 1 of [30] for SOS, and Table II of [29] for FALCm method, and the RMSD's obtained adopting the best settings for JMf. We do not compare the computational time as they are not provided in the above references. It can be noted that our results are in line with those produced by the other systems, even if a general fragment database has been used in our system.

Loop	**Average RMSD**			
Length	CCD	SOS	FALCm	JMf
4	0.56	0.20	0.22	0.30
8	1.59	1.19	0.72	1.31
12	3.05	2.25	1.81	1.97

Table 1: Comparison of loop sampling methods

7 Conclusions

In this paper, we presented a novel constraint (joined-multibody) to model rigid bodies connected by joints, with constrained degrees of freedom in the 3D space. We presented a proof of NP-completeness of the joined-multibody constraint, a filtering algorithm that exploits the geometrical features of the rigid bodies and showed its application in sampling protein loop conformations. Feasibility of the method is shown by performing loop reconstruction tests on a set of loop targets, with lengths ranging from 4 to 12 amino acids. The search space of the protein loop conformations generated is reduced with a controlled loss of quality.

The propagator has been presented as a filtering algorithm based on a directional growth of the rigid body. As future work, we plan to develop a "bi-directional" search that starts from both end anchors. A preliminary study shows that the error propagation, due to the clustering relation, can be bounded with greater accuracy. On the theoretical side, we are interested in proving the NP-hardness in the case of JM constraint based on compatible multi-bodies.

Following the loop conformation direction, we also plan to tune the fragment database and to integrate our filtering algorithm method with other refinements strategies to eliminate infeasible physical loop conformations, (e.g. *DFIRE* potential [37]), to increase the quality of the loop predictions.

As future work, we also plan to apply our filtering method to other related applications: protein-protein interaction, protein flexibility and docking studies. Those systems can be modeled by a set of joined-multibodyconstraints and it appears promising the possibility to explore a large set of conformations by representatives enumeration only.

Acknowledgments. The research has been partially supported by a grant from the Army High Performance Computing Research Center, NSF grants CBET-0754525 and IIS-0812267, and PRIN 2008 (20089M932N).

References

1. R. Backofen and S. Will. A Constraint-Based Approach to Fast and Exact Structure Prediction in 3-Dimensional Protein Models. *Constraints*, 11(1):5–30, 2006.
2. P. Barahona and L. Krippahl. Constraint programming in structural bioinformatics. *Constraints*, 13(1-2):3–20, 2008.
3. M. Ben-David, O. Noivirt-Brik, A. Paz, J. Prilusky, J. L. Sussman, and Y. Levy. Assessment of CASP8 structure predictions for template free targets. *Proteins*, 77:50–65, 2009.
4. M. Best, K. Bhattarai, F. Campeotto, A. D. Palú, H. Dang, A. Dovier, F. Fioretto, F. Fogolari, T. Le, and E. Pontelli. Introducing FIASCO: Fragment-based Interactive Assembly for protein Structure prediction with COnstraints. In *Proc. of Workshop on Constraint Based Methods for Bioinformatics*. `http://www.dmi.unipg.it/WCB11/wcb11proc.pdf`, 2011.
5. S. Cahill, M. Cahill, and K. Cahill. On the kinematics of protein folding. *Journal of Computational Chemistry*, 24(11):1364–1370, 2003.
6. A. Canutescu and R. Dunbrack. Cyclic coordinate descent: a robotics algorithm for protein loop closure. *Protein Sci*, 12:963–972, 2003.
7. Y. Choi and C. M. Deane. FREAD revisited: Accurate loop structure prediction using a database search algorithm. *Proteins*, 78(6):1431–40, May 2010.

8. A. Dal Palù, A. Dovier, and F. Fogolari. Constraint logic programming approach to protein structure prediction. *BMC Bioinformatics*, 5:186, 2004.
9. A. Dal Palù, A. Dovier, F. Fogolari, and E. Pontelli. CLP-based protein fragment assembly. *TPLP*, 10(4-6):709–724, 2010.
10. A. Dal Palù, A. Dovier, F. Fogolari, and E. Pontelli. Exploring Protein Fragment Assembly Using CLP. In T. Walsh, editor, *IJCAI*, pages 2590–2595. IJCAI/AAAI, 2011.
11. A. Dal Palù, A. Dovier, and E. Pontelli. A constraint solver for discrete lattices, its parallelization, and application to protein structure prediction. *Softw., Pract. Exper.*, 37(13):1405–1449, 2007.
12. A. Dal Palù, A. Dovier, and E. Pontelli. Computing approximate solutions of the protein structure determination problem using global constraints on discrete crystal lattices. *IJDMB*, 4(1):1–20, 2010.
13. A. Dal Palú, F. Spyrakis, and P. Cozzini. A new approach for investigating protein flexibility based on Constraint Logic Programming: The first application in the case of the estrogen receptor. *European Journal of Medicinal Chemistry*, 49:127–140, 2012.
14. C. Deane and T. Blundell. CODA. A combined algorithm for predicting the structurally variable regions of protein models. *Protein Sci*, 10:599–612, 2001.
15. I. Dotú, M. Cebrián, P. Van Hentenryck, and P. Clote. On Lattice Protein Structure Prediction Revisited. *IEEE/ACM Trans. Comput. Biology Bioinform*, 8(6):1620–1632, 2011.
16. A. Felts, E. Gallicchio, D. Chekmarev, K. Paris, R. Friesner, and R. Levy. Prediction of protein loop conformations using AGBNP implicit solvent model and torsion angle sampling. *J Chem Theory Comput*, 4:855–868, 2008.
17. A. Fiser, R. Do, and A. Sali. Modeling of loops in protein structures. *Protein Sci*, 9:1753–1773, 2000.
18. F. Fogolari, L. Pieri, A. Dovier, L. Bortolussi, G. Giugliarelli, A. Corazza, G. Esposito, and P. Viglino. Scoring predictive models using a reduced representation of proteins: model and energy definition. *BMC Structural Biology*, 7(15):1–17, 2007.
19. Y. Fujitsuka, G. Chikenji, and S. Takada. SimFold energy function for de novo protein structure prediction: consensus with Rosetta. *Proteins*, 62:381–398, 2006.
20. R. Hartenberg and J. Denavit. A kinematic notation for lower pair mechanisms based on matrices. *Journal of applied Mechanics*, 77:215–221, 1995.
21. M. Jacobson, D. Pincus, C. Rapp, T. Day, B. Honig, D. Shaw, and R. Friesner. A hierarchical approach to all-atom protein loop prediction. *Proteins*, 55:351–367, 2004.
22. M. Jamroz and A. Kolinski. Modeling of loops in proteins: a multi-method approach. *BMC Struct. Biol.*, 10(5), 2010.
23. R. Jauch, H. Yeo, P. R. Kolatkar, and N. D. Clarke. Assessment of CASP7 structure predictions for template free targets. *Proteins*, 69:57–67, 2007.
24. D. Jones. Predicting novel protein folds by using FRAGFOLD. *Proteins*, 45:127–132, 2006.
25. K. Karplus, R. Karchin, J. Draper, J. Casper, Y. Mandel-Gutfreund, M. Diekhans, and R. H. Source. Combining local structure, fold-recognition, and new fold methods for protein structure prediction. *Proteins*, 53(6):491–497, 2003.
26. L. Kinch, S. Yong Shi, Q. Cong, H. Cheng, Y. Liao, and N. V. Grishin. CASP9 assessment of free modeling target predictions. *Proteins*, 79:59–73, 2011.
27. S. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.
28. J. Lee, S. Kim, K. Joo, I. Kim, and J. Lee. Prediction of protein tertiary structure using profesy, a novel method based on fragment assembly and conformational space annealing. *Proteins*, 56(4):704–714, 2004.
29. J. Lee, D. Lee, H. Park, E. Coutsias, and C. Seok. Protein Loop Modeling by Using Fragment Assembly and Analytical Loop Closure. *Proteins*, 78(16):3428–3436, 2010.
30. P. Liu, F. Zhu, D. Rassokhin, and D. Agrafiotis. A self-organizing algorithm for modeling protein loops. *PLOS Comput Biol*, 5(8), 2009.
31. C. S. Rapp and R. A. Friesner. Prediction of loop geometries using a generalized born model of solvation effects. *Proteins*, 35:173–183, 1999.
32. M. Shen and A. Sali. Statistical potential for assessment and prediction of protein structures. *Protein Sci*, 15:2507–2524, 2006.
33. A. Shmygelska and H. Hoos. An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem. *BMC Bioinformatics*, 6, 2005.
34. K. Simons, C. Kooperberg, E. Huang, and D. Baker. Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and bayesian scoring functions. *J. Mol. Biol.*, 268:209–225, 1997.
35. V. Spassov, P. Flook, and L. Yan. LOOPER: a molecular mechanics-based algorithm for protein loop prediction. *Protein Eng*, 21:91–100, 2008.
36. Z. Xiang, C. Soto, and B. Honig. Evaluating conformal energies: the colony energy and its application to the problem of loop prediction. *PNAS*, 99:7432–7437, 2002.
37. H. Zhou and Y. Zhou. Distance-scaled, finite ideal-gas reference state improves structure-derived potentials of mean force for structure selection and stability prediction. *Protein Sci*, 11:2714–2726, 2002.