Designing succinct structural alphabets

Li, Shuai Cheng; Bu, Dongbo; Gao, Xin; Xu, Jinbo; Li, Ming

Published in:
Bioinformatics

Published: 01/07/2008

Document Version:
Final Published version, also known as Publisher's PDF, Publisher's Final version or Version of Record

Published version (DOI):
10.1093/bioinformatics/btn165

Publication details:
Li, S. C., Bu, D., Gao, X., Xu, J., & Li, M. (2008). Designing succinct structural alphabets. *Bioinformatics*, *24*(13). https://doi.org/10.1093/bioinformatics/btn165

BIOINFORMATICS

Vol. 24 ISMB 2008, pages i182–i189
doi:10.1093/bioinformatics/btn165

Designing succinct structural alphabets

Shuai Cheng Li [1], Dongbo Bu [1,2], Xin Gao [1], Jinbo Xu [3,*] and Ming Li [1,*]

[1]David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ont. N2L 3G1, Canada, [2]Institute for Computing Technology, Chinese Academy of Sciences, China and [3]Toyota Technological Institute at Chicago, IL 60637, USA

ABSTRACT

Motivation: The 3D structure of a protein sequence can be assembled from the substructures corresponding to small segments of this sequence. For each small sequence segment, there are only a few more likely substructures. We call them the 'structural alphabet' for this segment. Classical approaches such as ROSETTA used sequence profile and secondary structure information, to predict structural fragments. In contrast, we utilize more structural information, such as solvent accessibility and contact capacity, for finding structural fragments.

Results: Integer linear programming technique is applied to derive the best combination of these sequence and structural information items. This approach generates significantly more accurate and succinct structural alphabets with more than 50% improvement over the previous accuracies. With these novel structural alphabets, we are able to construct more accurate protein structures than the state-of-art *ab initio* protein structure prediction programs such as ROSETTA. We are also able to reduce the Kolodny's library size by a factor of 8, at the same accuracy.

Availability: The online FRazor server is under construction

Contact: {scli,mli}@uwaterloo.ca, j3xu@tti-c.org

1 INTRODUCTION

A small amount of structural fragments can model protein structures accurately (Kolodny *et al.*, 2002). Thus building such structural fragment libraries has attracted intensive research. The library size and accuracy are dominating factors for modelling and predicting the protein structures accurately. Compact independent libraries for protein structures have been built, and it is difficult to reduce the size of such libraries further. However, a sequence segment does not adopt all the structural fragments in a library with equal probabilities. Therefore, it is more reasonable to build a customized structural candidate list for each sequence segment. This way, the size of structural candidate list can be much more succinct, and the protein structure can be modelled more accurately.

1.1 Fragment libraries

Structural fragment libraries are also referred to as 'structural alphabet' in literature. The size of a fragment library may vary from dozens to hundreds. The fragments may have fixed or variable lengths. Typically, fragments in these libraries have lengths not more than nine since the structure database may not contain representative resemblances for longer fragments (Fidelis *et al.*, 1994).

Kolodny *et al.* (2002) studied fragment library with k-means clustering methods and showed that it is unnecessary to have a large fragment library to accurately model protein structures and construct near native structures. In that paper, fragments with lengths 4–7 were built with library sizes varying from 4 to 250. Criteria of evaluating fragment libraries for building protein structures were studied in Holmesand and Tsai (2004). Besides extracting structural fragments from known proteins, research has also been conducted on constructing structural fragments with *ab initio* methods, and such methods produced longer fragments (Lovell *et al.*, 2003).

For the protein structure prediction purpose, it is more desirable to have a position-specific structural fragment list for every sequence segment of the target. Only a limited number of structural fragments in the fragment libraries can be adopted as candidate structural fragment for a sequence segment. ROSETTA implemented this idea based on only two type of information: sequence profile and secondary structure (Rohl *et al.*, 2004; Simons *et al.*, 1997). Specifically, sequence profiles for the query sequence and each sequence in the structure database are generated by PSI-BLAST (Altschul *et al.*, 1997). A profile–profile similarity score between a query sequence segment and a structural fragment is calculated using a distance function called City Block Metric (CBM):

$$\text{DISTANCE} = \sum_{i=1}^{\ell} \sum_{aa=1}^{20} |S(aa,i) - X(aa,i)| \qquad (1)$$

where ℓ is the fragment length, $S(aa,i)$ and $X(aa,i)$ are the frequencies of amino acid aa at position i in the sequence segment and in the structural fragment, respectively. In addition, for a query sequence segment, its predicted secondary structure is compared with the known secondary structure of each structural fragment.

Unlike ROSETTA with fixed fragment lengths, TASSER (Zhang, 2007), extracts the structural fragments from structural models generated by threading programs, with variable fragment length.

1.2 Fragment-based protein structure prediction

The structure predictions based on fragment assembly have shown promising results. For example, two top automatic methods in CASP7 (Moult *et al.*, 2005), ROSETTA and TASSER, both use fragment assembly strategy. Fragment-based protein structure prediction is done in two steps: (1) identify the building blocks, which are fragments of known structures; (2) construct the protein structure with those building blocks using some search or simulation algorithms.

Fragment-based protein structure prediction method can be traced back to Pauling and Corey (1951), in which a protein fold is simplified into smaller parts by using the regular secondary structure element prediction. Research intensified after the work of

*To whom correspondence should be addressed.

Jones and Thirup (1986), which uses known structures to refine structures. Various fragment based structure prediction methods were developed in Jones and Thirup (1986), Claessens *et al.* (1989), Unger *et al.* (1989), Simon *et al.* (1991), Levitt (1992), Sippl (1993), Wendoloski and Salemme (1992), Bowie and Eisenberg (1994), with some success. Later, ROSETTA was developed, (Bradley *et al.*, 2003; Chivian *et al.*, 2005; Simons *et al.*, 1997). ROSETTA significantly improved the protein structure prediction based on fragments. Some recent fragment assembly algorithms, including Haspel *et al.* (2003), Inbar *et al.* (2003) and Lee *et al.* (2005), use longer fragments and/or different simulation algorithms. In another related work, De Brevern *et al.* (2004) proposed a 16-letter alphabet to predict local structures of a protein.

1.3 Our contributions

A major bottleneck for the fragment-based protein structure prediction methods is designing succinct and highly accurate structural alphabet. A constant factor reduction on the library size will result in an exponential reduction of the search space. We have introduced new ideas and have demonstrated that

- introducing structural information items, such as secondary structure, solvent accessibility and contact capacity, can improve the prediction of structural fragments;
- by using integer linear programming, we can derive the best combination of both sequence and structural information items, and it is possible to significantly reduce the structural alphabet (or library) size, at the same level of accuracy;
- such reduction will indeed significantly improve the protein structure prediction, with all other conditions unchanged.

The Occam's Razor principle tells us that smaller the alphabet is, the more likely it produces the right structure. A software package, FRazor, 'F' for fragment, based on integer linear programming is developed. By comparing our FRazor to the ROSETTA's fragment selection method, with the threshold as 1 Å, and fragment length 9, the position coverage is improved from 56.4% to 79.1% for β-sheets, and from 55.5% to 67.9% for loops while reducing the candidate list size from 25 to 10 simultaneously. With the candidate list size remaining at 25, our method can improve the position coverage of β-sheets from 56.4% to 89.6% and the position coverage of loops from 55.5% to 78.1%. The improvement of our method is over 50% for β-sheets and loops on average over the previous accuracies. Even for α-helices, where the improvement space is very little, FRazor still improves about 20% of the remaining open gap. Applying our method to Kolodny's library, our method reduces its size by a factor of 8 while achieving the same accuracy. Applying our fragment selection method to ROSETTA, with exactly the same settings for the rest of the system, we improve the prediction accuracy from 2% to 26% for 5 out of 6 standard benchmark proteins used in Simons *et al.* (1997), Kolodny *et al.* (2002) and Hamelryck *et al.* (2006).

In our work Li *et al.* (2007), we utilize structural fragment libraries in a new method for protein structure prediction. Experimental results show that structural fragments serve as a solid foundation for local structural bias prediction. In this article, we focus on generating high-quality structural fragment libraries.

2 METHODS

2.1 Problem statement

Given a protein target sequence t of length n, we parse t into a collection of sequence segments. We use a sliding window of a fixed length ℓ and step size 1 to parse t. Let these segments be $qe^1, qe^2, \ldots, qe^p$, $p = n - \ell + 1$ (qe stands for *query sequence element*), and denote the *native structural* fragments of these segments as $ns^1, ns^2, \ldots, ns^p$.

We need to have a collection of structural fragments from which we can select the structural candidates for sequence segments. Denote this collection of structural fragments as (se in below stands for *structural element*): $\mathcal{S} = \{se^1, se^2, \ldots, se^q\}$.

We refer to this collection of structural fragments as *structural space*. In this article, $\mathcal{S}$ is obtained by parsing the 40 protein structures listed in Table 1A.

We wish to select some structural fragments for each sequence segment, such that our selections contain adequate structural fragments close to the native structure of the sequence segment. Intuitively, the more the native-like structural fragments, the better decoys can be constructed.

Stated formally, given qe^j, $1 \le j \le p$, integer k and k', $k' \le k$ and a distance threshold θ, we wish to select a set of structural fragments, denoted as $\mathbb{S}_j \subset \mathcal{S}$, such that:

- $|\mathbb{S}_j| = k$;
- $\exists F_j \subset \mathbb{S}_j$ with $|F_j| \ge k'$ and
- $\forall s \in F_j$, $dist(s, ns^j) \le \theta$. $dist$ is a given distance function.

F_j is referred to as a subset of *near native* structures for qe^j. If F_j is non-empty, we say that qe^j is *covered* by $\mathbb{S}_j$. Using the 40 proteins in Table 1A to generate $\mathcal{S}$, we have verified that over 99% sequence positions (over all protein sequences) are covered by $\mathcal{S}$. $\mathbb{S}_j$ is referred to as the *structural candidate list*, or simply the *candidate list*, of qe^j. $\mathbb{S}_j$ is the 'alphabet' for qe^j.

2.1.1 Structural distance criteria To compute the distance between structural fragments, we use the standard measure, which is the backbone Cα-carbon root-mean-squared deviation (or CαRMSD). CαRMSD, or simply RMSD, satisfies the triangle inequality for fragments of equal length. Our methods are also applicable to other distance measures.

2.1.2 Structural space We generated the structural space from the 40 proteins in Table 1A, selected from the PDB. We observed that over 99% of the sequence/structural segments from the CASP7 proteins are covered by the structural fragments from these proteins.

ROSETTA and TASSER use similar approaches, selecting the fragments directly from the PDB (Berman *et al.*, 2000). It is also possible to use the existing independent libraries, such as Kolodny's fragment library (Kolodny *et al.*, 2002). It is not important which method to use, so long as we ensure that any structural fragment can find at least one resemblance in our structural space within some distance threshold.

2.2 Generalized linear model

It is reasonable to assume that introducing more structural information will help structural fragment prediction. However, how to combine these information items remains a difficulty. In this article, we design an integer linear programming model to integrate both sequence and structural information items optimally.

Between each structural fragment se^i in the structural space and each sequence segment qe^j, a feature vector, $\mathcal{V}^{i,j} = \langle v_1^{i,j}, \ldots, v_d^{i,j} \rangle$, of length $d = 4 \times 9$, will be computed to measure how well se^i and qe^j match, for $1 \le i \le q$ and $1 \le j \le p$. Each entry in $\mathcal{V}^{i,j}$ may be a linear or non-linear scoring function. We label $\mathcal{V}^{i,j}$ with $+1$ if $dist(se^i, ns^j) \le \theta$, and -1 otherwise.

Immediately, we may employ traditional machine learning approaches, such as support vector machines (SVMs), to classify $\mathcal{V}^{i,j}$ into two classes: class $+1$ contains the feature vectors labelled with $+1$ and class -1 contain

the feature vectors labelled with -1. Then the set of se^i's whose $\mathcal{V}^{i,j}$ is labelled $+1$ can be treated as the candidate list for qe^j. However, such an approach is too simplistic for our case. We do not really need to classify all the structural elements correctly. As a matter of fact, we do not care if most of the structural elements are classified wrongly. We only need at least one (or a few) of the structural fragments for qe^j classified correctly. Often there are close-to-native structural fragments for a particular sequence segment, we are just interested in selecting a candidate list of size say $k=25$, and one of them is native like. Furthermore k is a constant much smaller than the total number of native-like structural fragments, hence we do not need to classify most of the structural elements correctly. It is still possible to design an SVM to classify subsets of k-elements with at least one element correct. However, this significantly increases learning dimension hence requiring more data. Furthermore, since the features are for individual elements, this approach is less natural than our direct customized approach.

On the other hand, our classification task can be easily modelled by a linear model, since a candidate list is separable by a hyperplane with high probability. To see this, let us treat the feature vectors as high-dimensional points. Suppose we have a set of random points, each point is labelled with $+1$ or -1, we want to find a subset of the point set, such that the subset contains at least one point with label $+1$. We first form the smallest convex hull containing all points. For each vertex of the convex hull, if it is labelled with $+1$, then we use a hyperplane to separate it from the rest. We are done. The probability that no vertex on the hull is labelled with $+1$ can be estimated as $1-(1-P)^{|H|}$, where P is the probability that a point is in class $+1$ and $|H|$ is the expected number of points on the convex hull. According to Sims and Kim (2006), $P\approx(1/1.6)^9=0.015$ for fragments of length 9. According to Efron (1965), we may assume $|H|$ is sufficiently large to make $1-(1-P)^{|H|}$ approach to 1. Thus, with high probability, a linear separator can be trivially obtained.

These two observations have inspired us to design a system of linear models to solve our problem.

2.2.1 A generalized linear model formulation. A general linear model has the form:

$$y(\mathbf{x},\mathbf{w})=w_0+\sum_{k=1}^{M}w_k\phi_k(\mathbf{x}) \tag{2}$$

where $\mathbf{w}=(w_0,...,w_M)^T$, $\mathbf{x}$ is the input data, and $(\phi_1,...,\phi_M)^T$ are the *basis functions*. Here, $\mathbf{w}$ is the *weight vector or the parameters* we want to train, and w_0 is called a bias parameter and used for any fixed offset in the data. The basis functions, the ϕ_k's, are generally non-linear and are applied to the original data variables. In a linear model, $y(\mathbf{x},\mathbf{w})$ is a non-linear function of the input variables due to the non-linearity of the basis functions. We refer readers to Bishop (2006) for comprehensive treatment of the linear models.

We now generalize the spirit of linear model, Equation (2), to design an integer linear program (ILP) to model our problem. As mentioned above, we use a feature vector $\mathcal{V}^{i,j}=\langle v_1^{i,j},...,v_d^{i,j}\rangle$ to measure the similarity between a structural fragment se^i and a sequence segment qe^j. Without loss of generality, we assume $-1\le v_l^{i,j}\le 1$. Each structural fragment se^i is associated with a weight vector $\mathcal{W}^i=\langle w_1^i,...,w_d^i\rangle$. The distance between a structural fragment se^i and a sequence segment qe^j is computed by the dot product between $\mathcal{W}^i$ and $\mathcal{V}^{i,j}$:

$$\mathcal{D}^{i,j}=\sum_{l=1}^{d}w_l^i v_l^{i,j} \tag{3}$$

Thus, $\mathcal{V}^{i,j}=\langle v_1^{i,j},...,v_d^{i,j}\rangle$ may be regarded as a set of basis functions, the ϕ_k's, and we wish to adjust the parameters $\mathcal{W}^i=\langle w_1^i,...,w_d^i\rangle$ so that for some n_j, where se^{n_j} is a 'native-like' structure for qe^j, $\mathcal{D}^{n_j,j}$ is ranked top k among other $\mathcal{D}^{i,j}$'s for $1\le i\le q$. Thus we actually have a system of pq linear models, for $1\le i\le q$ and $1\le j\le p$. Also note that we only train one set of $\mathcal{W}^i=\langle w_1^i,...,w_d^i\rangle$ for each structural element se^i, and eventually we require only one of the $\mathcal{D}^{i,j}$'s, $1\le i\le q$, which is a native-like structure for qe^j, to be ranked well.

This model is generic. Although we assume a linear combination of the features, we do not assume any linearity about $v_l^{i,j}$'s, and they can contain quadratic terms and so on. For example, in Equation (1), a feature vector with length 180 is used. Each structure or sequence segment of length 9 is represented by 9×20 frequency distribution matrices. The feature vector has a size 180, and each entry is the absolute value of the difference between the corresponding entries. Here the $\mathcal{V}^{i,j}$ values are pre-calculated.

Now, our task is to train the $\mathcal{W}^i$'s. We used the 30 protein sequences, whose structures are known, in Table 1B as the Training Set. We parsed these proteins, length 9, Step 1, to obtain the set of qe^j's. The Structure Space se^i's are obtained from the 40 proteins, also with known structures, in Table 1A.

For each sequence segment qe^j, we pre-compute $\mathcal{Q}^j$, the set of structural fragments which have a distance to qe^j's native structure less than the distance threshold θ. Our objective here is to optimize the weights, the W^i's, of the distance function such that we have a distance function that ranks at least k' element in $\mathcal{Q}^j$ well. In the following formulation, we use $k'=1$ for the simplicity of presentation. We can easily extend the above model to the case such that in each candidate lists, at least k', $1\le k'\le k$, native-like structures are included.

For $1\le i\le q$, indexing the structural space and $1\le j\le p$, indexing the sequence segments, the ILP is as follows:

$$\min\sum_{j=1}^{p}g_j \tag{4}$$

$$\mathcal{D}^{n_j,j}-\mathcal{D}^{i,j}\le d_{n_j,i,j}(2+\epsilon)-\epsilon, n_j\in\mathcal{Q}^j, i\notin\mathcal{Q}^j, \forall j \tag{5}$$

$$\sum_{1\le i\le q, i\notin\mathcal{Q}^j} d_{n_j,i,j}\le k-1+f_{n_j,j}(q-(k-1)), n\in\mathcal{Q}^j, \forall j \tag{6}$$

$$\sum_{n_j\in\mathcal{Q}^j} f_{n_j,j}\le|\mathcal{Q}^j|-1+g_j, \forall j \tag{7}$$

$$\sum_{l=1}^{d}w_l^j\le 1, \forall j \tag{8}$$

$$d_{n_j,i,j}, f_{n_j,j}, g_j\in\{0,1\}, w_i^j\in[0,1]$$

We now interpret the above ILP formulation.

- Variable $g_j=1$ indicates no element in $\mathcal{Q}^j$ is included as one of the k elements for $\mathbb{S}_j$. The objective of the ILP, Equation (3), is therefore to minimize $\sum_{j=1}^{p}g_j$.
- The constant ϵ in Equation (4) is created as a gap to separate the native and non-native-like structural fragments. Ideally we want to have the parameter settings such that:
$$\mathcal{D}^{n_j,j}+\epsilon\le\mathcal{D}^{i,j} \tag{9}$$
where se^{n_j} is a native-like structure for qe^j, and se^i is a non-native-like structure for qe^j. Equation (4) is used to achieve this goal. It is clear that $-2\le\mathcal{D}^{n_j,j}-\mathcal{D}^{i,j}\le 2$. If $d_{n_j,i,j}=0$, we rank the near native-like structure se^{n_j} better than the non-native-like structure se^i.
- Equation (5) intends to check if a native-like structure se^{n_j} is in the candidate list of size k. If $f_{n_j,j}=0$, then the number of non-native-like structural fragments for qe^j that scores higher than $se^{n_j}\in\mathcal{Q}^j$ is less than k. When this fails, $f_{n_j,j}=1$.
- If $g_j=0$, Equation (6) ensures that at least one near native structure in $\mathcal{Q}^j$ for sequence segment qe^j is in its candidate list. The objective function Equation (3) is used to minimize the number of sequence segments whose candidate list of size k does not contain a near native structure. Equation (7) is just to normalize the parameter distributions.

2.3 Basis functions: the $V^{i,j}$'s

The basis functions we use in this article are the entries extracted from the sequence and structural information. Specifically, $4\times\ell$ entries, the $v_l^{i,j}$'s, are

created for each structural fragment and sequence segment pair. That is, for each se^i and qe^j pair, the $\mathcal{V}^{i,j}$ feature vector has $d=36$ entries (i.e. the ϕ_k's), with four entries, corresponding to the four types of scores below, for each position.

For simplicity, in this section, we denote a structural fragment as *se*, and denote a sequence segment as *qe*, without superscripts. They both have length $\ell=9$. The i-th positions of *qe* and *se* are denoted as $qe[i]$ and $se[i]$, respectively. For each position i, the following four types of score entries are created.

2.3.1 Mutation scores Mutation score is similar to that of ROSETTA, as shown in Equation (1), which is to compute the similarity score between profiles. The profiles for both the template and the sequence are obtained from 5-rounds of PSI-BLAST with a cutoff of 9×10^{-4}. Mutation score between *se* and *qe* consists of ℓ entries. One entry is calculated for each corresponding pair of positions. The value at position i, $1\le i\le\ell$ is defined to be:

$$\sum_{aa=1}^{20} S(aa,i)\times\log\frac{X(aa,i)}{S(aa,i)} \tag{10}$$

where we recall from Equation (1) that $S(aa,i)$ and $X(aa,i)$ are the frequencies of amino acid *aa* at position *i* for sequence segment and structural fragment, respectively. We have tested other possibilities, such as the City Block Metric, dot product and the one from Kim *et al.* (2003). We found that Equation (**??**) is slightly more stable.

2.3.2 Secondary structure score This term is to measure the similarity between the secondary structure of se^j and that of qe^j. The secondary structure for a structural element se^j is computed by DSSP, (Kabsch and Sander, 1983). The secondary structure of a sequence is predicted by PSIPRED (Jones, 1999), then it is parsed into qe^j's. The program predicts the confidences α_i, β_i and l_i for position i to be α-helix, β-sheet and loop, respectively.

The secondary structure score computation at position *i* is from Xu (2005):

- If the secondary structure type of $se[i]$ is α-helix, then we use α_i-l_i
- If the secondary structure type of $se[i]$ is β-sheet, then we use β_i-l_i
- If it is loop, we just use 0.

2.3.3 Contact capacity score For each structural position $se[i]$, a contact number n_i is calculated. There is a contact between two residues if the distance between their C_β atoms is within a given cutoff 7 Å. Contact capacity is to measure the capacity that a residue has c contacts with any other residues in a protein.

Given a protein structure, let $N(aa,c)$ be the number of residues with type *aa* and c contacts, $N(c)$ be the total number of residues having c contacts, $N(aa)$ be the number of residues with type *aa* and N be the total number of residues. Then for an amino acid type *aa*, the capacity to have c contacts is defined to be:

$$CC(c,aa)=-\log\frac{N\times N(aa,c)}{N(c)N(aa)}$$

The contact capacity score for position i is computed as: $\sum_{aa=1}^{20}S(aa,i)\times CC(n_i,aa)$.

2.3.4 Environmental fitness score The environment for each structural position is defined by the combination of secondary structure type and solvent accessibility. Three secondary structure types are used: α-helix, β-strand or loop; and three accessibility levels are defined: buried, intermediate and accessible. So in total there are nine states of the structural environment and each structural position has one of the nine environmental states. Let $F(E_i,aa)$ be the fitness score for an amino acid *aa* in environment state E_i. The fitness score between $se[i]$ and $qe[i]$ is calculated as: $\sum_{aa=1}^{20}S(aa,i)\times F(E_i,aa)$. For more details, we refer the readers to Kim *et al.* (2003).

3 RESULTS

We have implemented FRazor with C++, on Linux. The ILP is implemented with the package CPLEX. Additionally, we built some heuristics into the program in the case that ILP cannot find an optimal solution within a reasonable amount of time.

3.1 Evaluation criteria

We use fragment coverage, local fit approximation and position coverage as three evaluation criteria.

One way to evaluate the significance of selected structural fragments for each target is to simply count the percentage of sequence segments covered by the structural candidate lists for a given structure distance threshold. This percentage is referred to as *fragment coverage*.

Local fit approximation is a criterion developed in Kolodny *et al.* (2002) to evaluate the quality of a fragment library. For each sequence segment, the most similar structure in terms of RMSD from the structural candidate list is calculated. Then we take the average of the RMSD values over the entire sequence segment as the *local fit score*.

However, a better approach for the protein prediction purpose is to count the number of positions 'correctly predicted' in a target t. By 'correctly predicting a position' we mean that at least one sequence segment containing the position is covered. The percentage of the positions which are correctly predicted is referred to as *position coverage* in this work. This criterion is also used by Simons *et al.* (1997). The positions are divided into three cases *α-helix, β-sheet, and Loop*. We evaluate the coverage for each type of positions.

First, we compare FRazor's score function with ROSETTA's CBM. Then we show that our program does a much better job in selecting structural candidates from a fragment library. Finally, we show that the decoys assembled from the fragments generated by our method have better quality than those from ROSETTA's fragments.

3.2 Dataset

Our dataset consists of three parts: (1) Structure Space; (2) Training Set and (3) Testing Set. The Structure Space is the collection of structural fragments from which we can select the candidate structural fragments for a sequence segment. Training set consists of the fragments used to compute our parameters. Testing set contains proteins for evaluating our method.

The proteins for Structure Space and Training Set are both from a non-homologous (<30% homology) list with resolution <2 Å, dated on March 26, 2006. The list of these proteins was created by the program PISCES (Wang and Dunbrack, 2003), and totally there are 3177 chains. We used the first 70 chains. The Structure Space is made from 40 protein chains as shown in Table 1, Panel A. We parse these proteins with a sliding window of size $\ell=9$ and step size 1. Totally there are 9658 residues. The resulting structural space consists of 9338 length-9 structural fragments. The training data consist of the other 30 chains, which are also shown in Table 1, Panel B. We also parse them into length-$\ell=9$ segments with sliding window of step size 1. Totally there are 6584 residues.

For the Testing Set, we use proteins from CASP7 which were created after April, 2006; there are in total 94 proteins. Also the Testing Set are parsed into segments of length $\ell=9$. The CASP7 test proteins do not share high sequence identity with proteins in PDB released before March 26, 2006, which contain proteins in our

Table 1. Proteins for the Structure Space and Training Set

A. Structure Space						
1ci4a	1zm8a	1j79a	1rlja	1zhva	1wlya	2a14a
2gc9a	1lg7a	1wkoa	1jfla	1t9ha	1lm5a	1kxoa
1xfia	1rqpa	1m15a	1z96a	1mla	1ail	1yksa
1q25a	1mj5a	2erba	2bsya	1lst	1g8aa	1wzca
1y9wa	1xkpc	1v4va	1se8a	1p9ha	1r17a	1qfta
1aol	1ju3a	1rsga	1atg	1s5aa		
B. Training Set						
1olra	2byca	1yb5a	1pbwa	1v0ea	1orva	1jb7b
2ftra	1fj2a	1fp2a	2foma	1xtta	1suua	1xuua
1w2wb	1viaa	1r9wa	1fj2a	1dmga	2ah5a	1tc5a
2az4a	1mzwb	1ef1c	1uvqc	1ikta	1xfsa	1zava
1vk5a	1oyga					

The first 4 letters is the PDB code. The 5th letter is the chain id, missing for single chains.

Table 2. Position coverage for CBM versus FRazor (FR)'s score function

	α-Helix		β-Sheet		Loop		Overall	
θ	CBM	FR	CBM	FR	CBM	FR	CBM	FR
0.5	94.2	95.1	10.0	37.6	26.6	38.7	49.4	55.1
1	98.2	98.6	56.4	89.6	55.5	78.1	72.2	88.2
1.5	99.7	99.7	89.3	98.2	81.3	93.3	89.9	96.7
2	100	100	99.7	99.8	96.9	98.9	98.6	99.4
2.5	100	100	99.9	99.9	99.7	99.7	99.8	99.8
3	100	100	100	100	99.9	100	99.9	100
3.5	100	100	100	100	100	100	100	100

Position coverage (%) is displayed. The first column θ (Å) is the native threshold. The fragment candidate list size (k) is 25. The fragment length is 9.

Table 3. Position coverage percentage (%) for CBM versus FRazor (FR) at threshold value 1Å

	α-Helix		β-Sheet		Loop		Overall	
k	CMB	FR	CMB	FR	CMB	FR	CMB	FR
5	90.5	96.6	34.2	65.6	40.3	59.8	60.7	75.1
10	97.2	97.5	42.4	79.1	46.1	67.9	65.1	81.5
15	97.8	99.3	49.5	82.1	50.6	70.5	68.6	85.0
20	98.1	98.0	53.6	85.1	53.5	73.0	70.8	86.4
25	98.2	98.6	56.4	89.6	55.5	78.1	72.2	86.4
30	98.3	98.7	59.9	90.8	57.4	79.6	73.6	88.2
35	98.5	98.8	61.5	92.0	58.5	81.1	74.5	90.0
40	98.7	99.0	63.5	92.9	59.5	82.3	75.4	90.8

The first column is the fragment candidate list size. The fragment length is 9.

Table 4. Fragment coverage and local fit score for threshold value as 1 Å

	Fragment coverage (%)		Local fit score (Å)	
k	CBM	FRazor	CBM	FRazor
5	29.2	37.9	1.860	1.542
10	33.1	43.3	1.592	1.338
15	35.5	46.8	1.468	1.240
20	37.0	49.6	1.393	1.176
25	38.2	51.5	1.342	1.133
30	39.3	53.2	1.301	1.097
35	40.1	54.6	1.272	1.072
40	40.8	55.6	1.247	1.050

k is fragment candidate list size. The fragment length is 9. The threshold value is 1 Å.

Structure Space and Training Set. We also used six test proteins that were used in previous studies in Simons *et al.* (1997), Kolodny *et al.* (2002) and Hamelryck *et al.* (2006) to compare the quality of their decoys assembled from FRazor's fragments with that of ROSETTA's fragments. These six test proteins are: Protein A (PDB code 1FC2), Homeodomain (1ENH), Protein G (2GB1), Cro repressor (2CRO), Protein L7/L12 (1CTF) and Calbindin (4ICB).

3.3 FRazor versus CBM

It is an interesting question whether structural information, such as secondary structure, solvent accessibility and contact capacity, can help the prediction of structural fragments. In this experiment, we explore this question by comparing FRazor against the CBM model (Simons *et al.*, 1997), where only sequence profile is used. The experimental results are listed in Table 2, where the fragment candidate list size is set to be 25, the number of templates used is 40, i.e. the 40 proteins in Table 1A, and the fragment length is 9.

Observe Table 2. With the threshold value as 0.5 Å, the position coverage increases from 10.0% to 37.6%, and from 26.6% to 38.7% for β-sheets and loops, respectively. With the threshold value as 1 Å, the position coverage increases from 56.4% to 89.6%, and 55.5% to 78.1% for β-sheets and loops, respectively. For threshold 1.5 Å, significant improvement is observed for β-sheets and loops as well. Overall, we can have a position coverage 88.2% and 96.7% for threshold value 1 Å and 1.5 Å , respectively, and the two values for CBM are 72.2% and 89.9%. While the improvement for α-helix looks small, because there is nothing much left to improve upon, FRazor still made 20% improvement over the remaining gap, for 0.5 Å and 1 Å.

In Table 3, we fix the threshold value as 1 Å and we compare the results by varying the candidate list size. The position coverage is displayed. The improvement for β-sheets is more than 30% on average with the same candidate list size. The improvement for loops is more than 20% on average for all the cases. From the table we can see that, the position coverage is increased from 56.4% to 79.1%, and from 55.5% to 67.9% for β-sheets and loops, respectively, while reducing the fragment candidate size from 25 to 10 simultaneously. By using 5 as the candidate list size, FRazor's performance is better than that of CBM with 40 as fragment candidate list size for β-sheets and loops. Also with using 15 as the candidate list size, FRazor's performance is better than CBM with 40 as the candidate list size in all the cases.

Table 4 shows the results of fragment coverage and local fit criteria. In Table 4, we fix the threshold value as 1 Å and we compare the results by varying the candidate list size. This table demonstrates that FRazor with candidate list size 10 has higher fragment coverage

Table 5. Customized fragment lists versus independent fragment libraries

	Fragment coverage (%)		Local fit score (Å)	
L or *k*	KFL	FRazor	KFL	FRazor
25	–	45.3	–	0.763
50	36.2	40.5	0.754	0.667
100	40.7	55.7	0.673	0.589
150	43.3	58.6	0.633	0.554
200	44.0	60.4	0.603	0.531
250	46.3	61.8	0.585	0.515

This first column is the fragment candidate list size for FRazor, and the library size for Kolodny's libraries. Fragment coverage (%) at threshold 0.5 Å is shown for Kolodny's fragment libraries (KFL) at Column 2 and for FRazor's distance function at Column 3, respectively.

Table 6. Decoy quality comparison between ROSETTA and FRazor

Target protein			ROSETTA			FRazor		
Name	L	α,β	%	Best	Avg	%	Best	Avg
1FC2	43	2,0	20.5	2.59	7.3	38.6	2.60	6.4
1ENH	54	2,0	39.5	3.06	7.3	53.8	2.61	6.4
2GB1	56	1,4	89.8	1.88	4.3	90.6	2.04	4.4
2CRO	65	5,0	40.6	3.02	6.7	67.2	2.57	5.8
1CTF	68	3,3	9.2	3.42	9.1	11.0	3.14	8.4
4ICB	76	4,0	2.8	4.74	9.4	2.6	4.81	9.6

The first column is protein name given as PDB code. *L* is sequence length. Third column is number of α-helices and β-stands. Column 4–6 give the percentage (%) of the good decoys with RMSD <6.0 Å, RMSD of the best decoy (Best), and average RMSD (Avg) of all decoys by ROSETTA. Column 7–9 give the corresponding values for FRazor.

than the fragment coverage of CBM with candidate list size 40, with scores 43.3% and 40.8%, respectively.

For all these evaluation criteria, we can safely draw a conclusion that FRazor is able to identify compact candidate lists for sequence segments. Besides the results reported, we conducted experiments on varying the fragment length and candidate list size. These experimental results suggest that FRazor is stable and robust, and consistent improvement is observed.

3.4 Selecting fragments from a library

Sequence-specific fragment candidate lists are able to model a protein more accurately than an independent fragment library. In this section, we show that FRazor can produce a more accurate fragment candidate list than an independent library by comparing to the fragment libraries from Kolodny *et al.* (2002). From another aspect that each structural fragment can be mapped to an entry in a fragment library, FRazor is able to select a subset of fragments from a library for a sequence segment. The libraries from Kolodny *et al.* (2002) with fragment length 7 are used, and the library sizes are 50, 100, 150, 200 and 250. In order to have a fair comparison, we re-evaluated the performances of these libraries on our test data. Denote the library size as *L*.

Table 5 shows the results of Kolodny library, and FRazor's customized lists. By using candidate list size 25, the fragment coverage score is better than the library with 200 fragments. The local fit score by using 100 fragments is comparable with a fragment library size 250.

3.5 Application to protein structure prediction

We also compared FRazor with ROSETTA's fragment generation module. This ultimate test is to examine the quality of the decoys folded from the fragments generated by FRazor. We replaced ROSETTA's fragment generation method by FRazor to test its accuracy. To fairly evaluate FRazor, we used ROSETTA's energy function and its default setting. The ROSETTA's fragment generation code is obtained from the ROSETTA package (version 2.0.1). For both FRazor and ROSETTA's fragment generation module, their structural fragments are selected from the same set of 40 proteins, which is included in ROSETTA's fragment generation module. Note that these are different 40 proteins from Table 1A. We used the same 30 proteins in Table 1B to train. Using FRazor instead of ROSETTA's fragment generation module, with everything else remain unchanged, we demonstrate that FRazor improves structure prediction accuracy significantly.

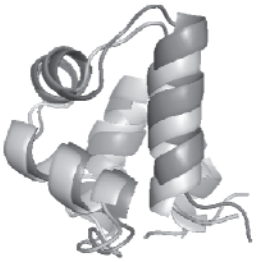
We used the six proteins that were used in previous studies (Hamelryck *et al.*, 2006; Kolodny *et al.*, 2002; Simons *et al.*, 1997) to evaluate FRazor. We assembled 1000 decoys for each protein using structural fragments generated from both FRazor and ROSETTA, respectively, and then compared FRazor and ROSETTA in terms of the percentage of *good decoys*[1] and the average RMSD of all the 1000 decoys. As shown in Table 6, FRazor can generate 1.8–26% more good decoys than ROSETTA's fragment generation method. The average RMSD of the decoys generated by FRazor is also much smaller for four of the six test proteins. For the other two test proteins, both FRazor and ROSETTA have similar average RMSD.

FRazor also generated the best decoys with better RMSDs. For example, the best decoy generated by FRazor for the Cro repressor protein (PDB code 2CRO) has a much lower RMSD to its native structure than that generated by ROSETTA. As shown in Figure 1, the first is the best decoy for the Cro repressor protein generated by ROSETTA with RMSD 3.02 Å, the second is the best decoy generated by FRazor with RMSD 2.57 Å, and the third is the native structure. In addition to the Cro repressor protein, the best decoys for both Homeodomain (PDB code 1ENH) and Protein L7/L12 generated (PDB code 1CTF) by FRazor also have much lower RMSDs than the best generated by ROSETTA. For the other three proteins 1FC2, 2GB1 and 4ICB, their best decoys generated by ROSETTA are slightly better than those by FRazor.

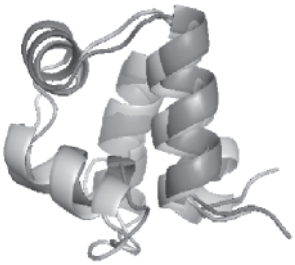
4 DISCUSSION

Our results illustrate that structural information can help the accurate prediction of structural fragments for a sequence segment. Our experimental results demonstrate that our method generates structural fragments of significantly better quality, compared to ROSETTA's fragment generator and Kolodny's library. This is further justified by the end results of decoys generation.

[1]A decoy is good if its RMSD to the native structure is less than 6 Å.



(**a**) The best decoy by ROSETTA (RMSD 3.02Å)



(**b**) The best decoy by FRazor (RMSD 2.57Å)

Fig. 1. Best decoys generated by ROSETTA and FRazor for the Cro repressor protein 2CRO.

We have generated our fragment library based on 40 proteins. Currently we are in the process of extending this to a much larger set of representative proteins. The scoring function we used to map a sequence segment to a structural fragment consists of mutation score, secondary structure score, contact capacity score, and environment fitness score. To improve the performance, a natural idea is to use more scoring terms. A promising way is to combine the scores from threading results, like the case in Zhang *et al.* (2005), Zhang (2007). Currently all the scoring terms depend on only a single position, which implies that residues in a protein sequence are assumed to be independent. However, some residues are obviously correlated, and it may obtain better performance if we encode the correlation information into our scoring function. The challenge to do so is to deal with the sparsity in training data since there will be many more parameters to be trained. We may use some regularization technique to resolve this issue.

Our work significantly improves the accuracy of β-sheet and loop positions, and this gives us the possibility of predicting loop regions more accurately. This work is underway. The program can also assign weights to the positions of a structure automatically. This might be useful for identifying structure motifs. A position with a small weight may imply this position is unstable.

ACKNOWLEDGEMENTS

We thank David Baker and his ROSETTA group, for developing and allowing us using the ROSETTA program, and ISMB'08 referees for helpful comments.

Funding: This work is supported by NSERC OGP0046506, the Canada Research Chair program and MITACS, and was made possible by the facilities of the SHARCNET (www.sharcnet.ca). D.B. was also partially supported by a Chinese Government Scholarship Program and an NSFC grant 60496320.

Conflict of Interest: none declared.

REFERENCES

Altschul,S.F. *et al.* (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.

Berman,H.M. *et al.* (2000) The protein data bank. *Nucleic Acids Res.*, **28**, 235–242.

Bishop,C.M. (2006) *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer.

Bowie,J.U. and Eisenberg,D. (1994) An evolutionary approach to folding small α-helical proteins that uses sequence information and an empirical guiding fitness function. *Proc. Natl Acad. Sci.*, **91**, 4436–4440.

Bradley,P. *et al.* (2003) Rosetta predictions in CASP5: successes, failures, and prospects for complete automation. *Proteins Struct. Funct. Genet.*, **53** (Suppl. 6), 457–468.

De Brevern,A. *et al.* (2004) Local backbone structure prediction of proteins. *In Silico Biol.*, **4**, 381–386.

Chivian,D. *et al.* (2005) Rosetta predictions in CASP5: successes, failures, and prospects for complete automation. *Proteins Struct. Funct. Genet*, **61** (Suppl. 7), 157–166.

Claessens,M. *et al.* (1989) Modelling the polypeptide backbone with 'spare parts' from known protein structures. *Protein Eng.*, **2**, 335–345.

Efron,B. (1965) The convex hull of a random set of points. *Biometrika*, **52**, 331–343.

Fidelis,K. *et al.* (1994) Comparison of systematic search and database methods for constructing segments of protein structure. *Protein Eng.*, **7**, 953–960.

Hamelryck,T. *et al.* (2006) Sampling realistic protein conformations using local structural bias. *PLoS Computat. Biol.*, **2**, e131.

Haspel,N. *et al.* (2003) Reducing the computational complexity of protein folding via fragment folding and assembly. *Protein Sci.*, **12**, 1177–1187.

Holmesand,J.B. and Tsai,J. (2004) Some fundamental aspects of building protein structures from fragment libraries. *Protein Sci.*, **13**, 1636–1650.

Inbar,Y. *et al.* (2003) Protein structure prediction via combinatorial assembly of sub-structural units. *Bioinformatics*, **19** (Suppl. 1), 158–168.

Jones,D.T. (1999) Protein secondary structure prediction based on position-specific scoring matrices. *J. Mol. Biol.*, **292**, 195–202.

Jones,T.A. and Thirup,S. (1986) Using known substructures in protein model building and crystallography. *EMBO J.*, **5**, 819–823.

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

Kim,D. *et al.* (2003) PROSPECT II: protein structure prediction program for genome-scale applications. *Protein Eng.*, **16**, 641–650.

Kolodny,R. *et al.* (2002) Small libraries of protein fragments model native protein structures accurately. *J. Mol. Biol.*, **323**, 297–307.

Lee,J. *et al.* (2005) Protein structure prediction based on fragment assembly and parameter optimization. *Biophys. Chem.*, **115**, 209–214.

Levitt,M. (1992) Accurate modeling of protein conformation by automatic segment matching. *J. Mol. Biol.*, **226**, 507–533.

Li,S.C. *et al.* (2007) FALCON: zero in on the native protein structure. *Technical Report*, University of Waterloo.

Lovell,S.C. *et al.* (2003) Ab initio construction of polypeptide fragments: efficient generation of accurate, representative ensembles. *Proteins*, **51**, 41–55.

Moult,J. *et al.* (2005) Critical assessment of methods of protein structure prediction (casp):round 6. *Proteins Struct. Funct. Genet.*, **61**, 3–7.

Pauling,L. and Corey,R.B. (1951) The pleated sheet, a new layer configuration of polypeptide chains. *Proc. Natl Acad. Sci.*, **37**, 251–256.

Rohl,C.A. *et al.* (2004) Protein structure prediction using Rosetta. *Methods Enzymol.*, **383**, 66–93.

Simon,I. *et al.* (1991) Calculation of protein conformation as an assembly of stable overlapping segments: application to Bovine pancreatic trypsin inhibitor. *Proc. Natl Acad. Sci.*, **88**, 3661–3665.

Simons,K.T. *et al.* (1997) Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and Bayesian scoring functions. *J. Mol. Biol.*, **268**.

Sims,G.E. and Kim,S.H. (2006) A method for evaluating the structural quality of protein models by using higher-order varphi-psi pairs scoring. *Proc. Natl Acad. Sci.*, **103**, 4428–4432.

Sippl,M. (1993) Recognition of errors in three-dimensional structures of proteins. *Proteins*, **17**, 355–362.

Unger,R. *et al.* (1989) A 3D building blocks approach to analyzing and predicting structure of proteins. *Proteins Struct. Funct. Genet.*, **5**, 355–373.

Wendoloski,J.J. and Salemme,F.R. (1992) Probit: a statistical approach to modeling proteins from partial coordinate data using substructure libraries. *J. Mol. Graph.*, **10**, 124–126.

Wang,G. and Dunbrack,R.L.,Jr. (2003) PISCES: a protein sequence culling server. *Bioinformatics*, **19**, 1589–1591.

Xu,J. (2005) Fold recognition by predicted alignment accuracy. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, **2**, 157–165.

Zhang,Y. (2007) Template-based modeling and free modeling by I-TASSER in CASP7. *Proteins*, (Suppl. 8).

Zhang,Y. *et al.* (2005) TASSER: an automated method for the prediction of protein tertiary structures in CASP6. *Proteins*, **61** (Suppl. 7), 91–98.