BIOINFORMATICS

Vol. 00 no. 00 2007
Pages 1–9

Designing Succinct Structural Alphabets

Shuai Cheng Li[a], Jinbo Xu[b*], Xin Gao[a], Dongbo Bu[a,c], Ming Li[*a]

[a]David R. Cheriton School of Computer Science University of Waterloo, Ontario, Canada N2L 3G1
[b]Toyota Technological Institute at Chicago, 1427 East 60th Street, Chicago, IL 60637
[c]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China 100080

ABSTRACT

Motivation: The 3D structure of protein sequence A can be assembled by the substructures corresponding to small segments of A. A sequence segment does not take on all the structural fragments and thus it is desirable to build a short customized structural candidate list for each sequence segment. For each sequence segment, these substructures are its "specific structural alphabet". The smaller these candidate lists are, the faster the protein structure can be constructed; the more accurate these candidate lists are, the more accurate the final protein structure will be. A major obstacle in protein structure prediction is to construct a small set of structural candidates for each segment such that the native structure can be rebuilt from these structural candidates accurately.

Results: Based on integer linear programming and incorporating extra structural information, a software package FragShaver is developed. We have made significant progress in overcoming the abovementioned obstacle:

- Comparing our package to the Rosetta's fragment selection method, at threshold 1Å and structural fragment length 9, the position coverage is improved from 56.4% to 79.1% for β-sheet, and from 55.5% to 67.9% for loop, while reducing the candidate list size from 25 to 10 simultaneously. By using candidate list size 25, our method improves Rosetta's position coverage of β-sheet from 56.5% to 89.6% and the position coverage of loop from 55.5% to 78.1%. At 1.5Å, we achieve position coverage 96.7%.
- Applying our method to Kolodny's independent library, our experiment indicates that our method is capable of identifying a small subset of structural candidates for a given sequence segment to achieve the same accuracry as using the whole library, reducing Kolodny's library size from 200 to 25 at the same accuracy.

Additionally, FragShaver is robust for optimizing the parameters of a distance function of structural fragment selection. This provides the specialists with an automatic tool for performing parameter optimization on a designed distance function for selecting the structural candidates.

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

1 INTRODUCTION

Protein structure modelling and determination are some of the fundamental tasks in molecular biology. Intensive research has been carried out for building structural fragment libraries to model and predict protein 3D structures. A small amount of structural fragments can model the protein structure accurately. The library size and accuracy are dominating factors for modelling and predicting the protein structures. Though the library size is small, from tens to hundreds, the size is still considered to be large for practical purposes such as protein structure prediction. Research has been conducted on building very compact independent libraries for protein structures. 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. Hence it is more reasonable to build a customized structural candidate list for each sequence segment. By this, the size of structural candidate list can be much more succinct, and the protein structure can be modelled more accurately. This work is essential to protein structure prediction and other related research.

1.1 Fragment Libraries

(Structural) Fragment libraries are also referred to as "structural alphabet" in literatures. The size of a fragment library may vary from tens to hundreds. The fragments may have fixed or variable lengths. The structural fragments may have all-atom descriptions or reduced descriptions with $C\alpha$s only. The libraries are used to predict or model protein structures (Rooman *et al.*, 1990; Fetrow *et al.*, 1997; Bystroff and Baker, 1998; Camproux *et al.*, 1999; de Brevern *et al.*, 2000, 2002; Hunter and Subramaniam, 2003; Camproux *et al.*, 2004; Etchebest *et al.*, 2005) and (Unger *et al.*, 1989; Levitt, 1992; Prestrelski *et al.*, 1992; Unger and Sussman, 1993; Schuchhardt *et al.*, 1996; de Brevern and Hazout, 2000; Micheletti *et al.*, 2000; Ashish V. Tendulkar and Wangikar, 2004).

The size of a library for prediction purpose is smaller than the size of a library for modelling purpose, as the latter requires a higher resolution. Fragments in these libraries typically have lengths no more than 9. The structure database may not contain representative ressemblances for longer fragments (Fidelis *et al.*, 1994). Kolodny *et al.* (2002) studied fragment library with k-mean clustering methods and showed that it is not necessary to have a large fragment library to accurately model protein structures and construct near native models. In that paper, fragments with length 4–7 were built with library sizes varying from 4 to 250. Holmes and Tsai (2004) studied the aspects and criteria of evaluating fragment libraries for building protein structures. 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 (DePristo *et al.*, 2003).

For protein structure prediction purpose, to have a position-specific structural fragment list for every sequence segment of the target is more desirable. Only a limited number of structural

*to whom correspondence should be addressed

© Oxford University Press 2007.

fragments in the fragment libraries can be adopted as candidate structural fragment for a sequence segment. In Simons *et al.* (1997); Rohl *et al.* (2004), each sequence segment is scored against each structural fragment of the same length (i.e. 9) and the structural fragments with smaller distance values are selected as candidates. The distant function is a simple City Block Metric:

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

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. The frequencies can be obtained typically from sequence alignment tools such as PSI-BLAST (Altschul *et al.*, 1997). Unlike Rosetta with fixed fragment lengths, Zhang (2006) extracts the structural fragments from the threading algorithms, and the lengths of the fragments are not fixed.

1.2 Fragment Based Protein Structure Prediction

The structure predictions based on fragment assembly has shown promising results. For example, two top methods in CASP7, Rosetta and Zhang-server, both use fragment assembly. 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); Pauling *et al.* (1951), in which a protein fold is simplified into smaller parts by using the regular secondary structure element prediction. Research intensified after Jones and Thirup (1986)'s work, which uses known structures to refine structures. 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) each developed fragment structure prediction methods and demonstrated success. Later, Rosetta was developed by Simons *et al.* (1997), which improved the protein structure prediction based on fragments significantly. Rosetta is still evolving, and keeps on producing promising results (Han *et al.*, 1997; Simons *et al.*, 1999; Bonneau *et al.*, 2001; Kuhlman *et al.*, 2003; Bradley *et al.*, 2003; Chivian *et al.*, 2005). Some recent fragment assembly algorithms, including Haspel *et al.* (2003); Inbar *et al.* (2003); Lee *et al.* (2005) use longer fragments and/or different simulation algorithms.

In the past two CASPs (Critical Assessment of Techniques for Protein Structure Prediction - biannual experiments to assess protein structure prediction methods (Moult *et al.*, 1999, 2001, 2003, 2005)), the top automatic server employed protein structural fragments. Rosetta, the best automatic server in CASP6 and second best in CASP7 selected length 9 structural fragments from known proteins as building blocks, and assembled them with a Monte Carlo search strategy directed by some statistical energy functions to yield native like structures. SP^3 (Zhou and Zhou, 2005), one of the best methods in CASP6, used fragments with similar structures to generate template profiles for threading purpose. (Zhang *et al.*, 2005; Zhang, 2006) developed TASSER, which generated various length fragments from threading and assembled them with a Monte Carlo method. For all of these methods, candidate structure selection for sequence segments serves as the foundation to obtain better predictions.

1.3 Our Contributions

For each sequence segment of a target, our method is capable of constructing a succinct list of structure candidates such that the native structure can be built from these structural fragments accurately.

A software package, FragShaver, based on integer linear programming is developed. By comparing our package to the Rosetta's fragment selection method, with the threshold as 1Å, and fragment length 9, the position coverage is improved from 56.4% to 79.1% for β-sheet, and from 55.5% to 67.9% for loop while reducing the candidate list size from 25 to 10 simultaneously. With the candidate list size as 25, our method can improve the position coverage of β-sheet from 56.5% to 89.6% and the position coverage of loop from 55.5% to 78.1%. Using the same amount of fragments, and with a threshold 1Å, the improvement of our method is 30% and 20% for β-sheet and loop on average. Applying our method to Kolodny's library, our experiment indicates that the method is potentially capable of identifying a small subset of fragments from the library for sequence segments to achieve high accuracy.

Our package is robust for optimizing the parameters of a distance function of structural fragment selection and allows specialists to have fast prototype of distance entries to select structural candidates.

2 APPROACH

We model the problem of tuning the parameters of a distance function as an optimization problem. We solve this optimization problem with integer linear programming (ILP). The ILP is guaranteed to find an optimal solution, which ensures that we find an optimal combination of the parameters for a distance function.

In our approach, each fragment structure adopts its own specific parameter settings. The intuition behind this is that the accuracy and usefulness of the distance entries are varying from structural fragments to structural fragments. By tuning the parameters, we can assign lower weights to the distance entries containing error or noise and assign higher weights for those entries which are more accurate.

We designed four types of scores for fragment selection: mutation score, secondary structure score, contact capacity score and environmental fitness score. The four types of scores are combined by using our ILP model.

Our system consists of two parts: (1) training the parameters; and (2) selecting candidate structural fragments for a sequence segment. The parameters for each structural fragment are identified with the training data set. To select the candidate structural fragments, each sequence segment is scored against each structural fragment. The structural fragments are ranked according to our distance function for each sequence segment. The top k structural fragments are selected as the candidate list for the sequence segment.

3 METHODS

Before we give the description of our methods for predicting structural fragments, we first formalize the problem. and the evaluation of experimental results.

3.1 Problem Statement

Given a protein target 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 in this paper. Let these fragments be $qe^1, qe^2, \ldots, qe^p$, $p = n - \ell + 1$, and denote the native structural fragments of these fragments as $ns^1, ns^2, \ldots, ns^p$.

We need to have a collection of structural fragments from which we can select the structure candidates for sequence segments. Denote this collection of structural fragments as:

$$\mathcal{S} = \{se^1, se^2, \ldots, se^q\}$$

We refer to this collection of structural fragments as *structural space* in this paper.

The problem that we want to address here is to identify some structural fragments for each sequence segment which contain at least one structural fragment which is close to the native structure of the sequence segment.

Stated formally: for qe^j, $1 \leq j \leq q$, and a given integer k and a distance threshold θ, we want to find a subset of structural fragments $\mathbb{S}_j \subset \mathcal{S}$, such that $|\mathbb{S}_j| \leq k$ and $\exists s \in \mathbb{S}_j$ with $dist(s, ns^j) \leq \theta$ for some distance function $dist$. If such an element s exists, we say that qe^j is *covered* by $\mathbb{S}_j$. It is clear that we assume any sequence segment is covered by $\mathcal{S}$. $\mathbb{S}_j$ is referred to as *structure candidate list* or simply *candidate list* and k is called *candidate list size* in this paper.

3.2 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 satisfies triangle inequalities for fragments of equal length. Our methods are also applicable to other distance measures.

3.3 Evaluation Criteria

We use fragment coverage (fc-score), local fit approximation (lf-score) and position coverage (pc-score) as the evaluation criteria.

One way to evaluate the significance of selected structural fragments for each target is to simply count 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 term of CαRMSD from the structure candidate list is used. Then we take the average this CαRMSD value over all the sequence segment as the *local fit score*.

However, a better approach for protein prediction purpose, is to count the number of positions "correctly predicted" in target t. By "correctly predicting a position" we mean that there is at least one sequence segment containing the position covered. The percentage of the positions which are correctly predicted is referred to as *position coverage* (pc) in this work. This criterion is also used by Simons *et al.* (1997) and it is important for protein prediction based on fragment assembly methods. The positions are divided into three cases *α-helix, β-sheet, and Loop*. We evaluate the coverage for each type of positions.

3.4 Data Set

Our data set consists of three parts: (1) Structure Database; (2) Training Set; and (3) Testing Set. The structure database 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.

Table 1. Proteins for Structural Database and Training Set

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

The first 4 letters is the PDB name, and the 5th letter is the chain id for each entry. If the 5th letter is a space, the whole protein is used in the structural space.

The structure database of fragments is made from 40 protein chains as shown in Table 1. We parse these proteins with a sliding window of size ℓ and step size 1. Totally there are 9,658 residues. The structural database consists of 9,338 length-9 structural fragments.

The training data consist of 30 chains, which are also shown in Table 1. We also parse them into length-ℓ fragments with sliding window of step size 1. Totally there are 6,584 residues.

The proteins for structure database and training set are both from a non-homologous (less than 30% homology) list with resolution $< 2\mathring{A}$, dated on March 26^{th}, 2006. The list of these proteins was created by the program PISCES (Wang and Roland L., 2003), and totally there are 3177 chains.

For the test data, we use proteins from CASP7 which were created after April, 2006; there are in total 94 proteins. Also the test set are parsed into fragments of length ℓ.

Our methods can be used for over varying fragment lengths. We choose fixed length mainly for the ease of comparison. Furthermore, the RMSD measure is not applicable across varying fragment lengths in general, and to compute lf-score over varying lengths is less reasonable. For example, a Cα RMSD threshold 2Å is considered to be accurate for a fragment with length 30, and unacceptable for a fragment with length 4.

3.5 Structural Space

For the structural space, we can use either existing libraries or employ the Rosetta/Zhang's approach, that is, to select the fragments directly from the PDB (Berman *et al.*, 2000). For a comprehensive fragment library, each structural element in the PDB can be mapped to some fragments in the library. This implies that a subset of fragments from the PDB corresponds to a subset of fragments from a fragment library and the two approaches are equivalent to some extend. In this paper, we choose to select the fragments directly from some proteins in the PDB.

Ideally, the structure space needs to ensure that any structural fragment can find at least one ressemblance in the space within some distance threshold. An independent library has size from tens to several hundreds. This implies that by using a small amount of proteins with known structures, most of the sequence segments can be covered.

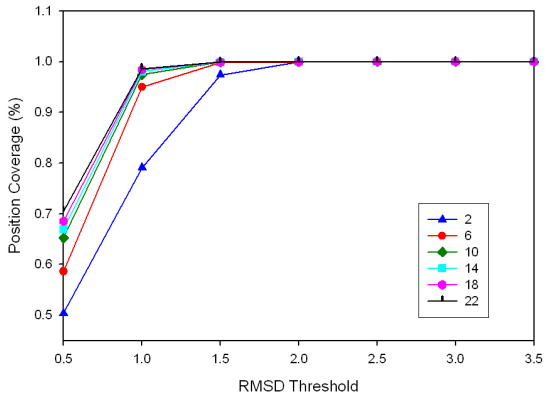

Fig. 1. Position Coverage vs. RMSD Threshold for Various Number of Templates. The fragment length is 9. The postion coverage is compute for the number of templates as: 2, 6, 10, 14, 18 and 22

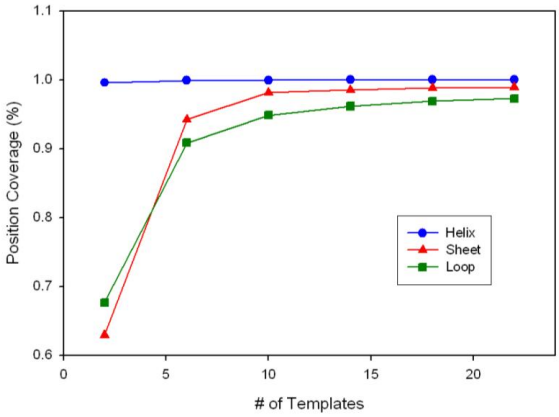

Fig. 2. Coverage for Helix, Sheet, and Loop vs. Number of Template. We use a distance threshold $1\mathring{A}$, and fragment length 9.

We conduct a simple study to show by using a small amount of proteins with known structures, most of the sequence segments can be covered by the structural fragments from these proteins.

Fig. 1 shows the relationship between the position coverage and distance threshold for different number of templates (or proteins with known structures), where the templates are randomly selected. Fragments with length 9 is used. The coverage increases steadily with the increment of the number of proteins. For a distance threshold of 2.5Å, even 2 templates are enough to cover more than 99% of the positions. For a distance threshold of 1.5Å, 6 templates or more are able to cover more than 99% of the positions. However, we note that for distance threshold $1\mathring{A}$, the coverage increases very slowly after more than 10 templates. With 18 templates, the position coverage values are more than 98% and 99.9% for distance thresholds 1Å and 1.5Å , respectively, which is considered to be accurate enough for the purpose of protein structure prediction. Also, we have compared the coverage with existing fragment libraries such as the ones from Kolodny *et al.* (2002). By using 40 templates as structural database, we can model protein structures more accurately than the independent libraries.

We ran the experiments on other selections of templates, and similar results are obtained. Also, as we increase the number of templates to 100, the coverage does not increase too much. We noticed that a small number of proteins do not imply that the structural space is small, as each template may contain several hundred structural fragments and it is still a very tedious task to select a customized candidate list with high coverage.

In Fig. 2, we set the distance threshold as 1Å. We divide the positions into 3 cases: α-helix, β-Sheet, and loop. By using more than 5 proteins, the α-helix positions have position coverage more than 99%. By using more than 15 proteins, the β-sheet positions and loop positions can have coverage more than 98% and 96%, respectively. The increasing rate of the coverage becomes very slow after the use of more than 15 proteins for both the β-sheet and the loop case. The figure also implies that the α-helix positions are more regular and we may be able to predict them accurately.

3.6 ILP Model for Parameter Settings

Some of the initial ideas are borrowed from J. Meller and R. Elber (2001). In that paper, linear programming is used to compute the profiles for threading.

We make the following assumption: between each structural fragment se^i and each sequence segment qe^j and , a feature vector with length d can be computed, and we denote it as: $\mathcal{V}^{i,j} = \langle v_1^{i,j}, \ldots, v_d^{i,j} \rangle$. Without loss of generality, we assume $-1 \leq v_l^{i,j} \leq 1$. Each structural fragment se^i is associated with a weight vector $\mathcal{W}^i = \langle w_1^i, \ldots, w_d^i \rangle$. The distance between a sequence segment and a structural fragment is computed by the dot product between $\mathcal{W}^i$ and $\mathcal{V}^{i,j}$, which is:

$$\mathcal{D}^{i,j} = \sum_{l=1}^{d} w_l^i v_l^{i,j} \tag{2}$$

This model is generic. Although we assume a linear combination of the feature, we do not assume any linearality about $\mathcal{V}^{i,j}$, and it can contain quadratic terms and so on. For example, in Eq. 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 absolution value of the difference between the corresponding entries.

For each sequence segment qe^j, $\mathcal{Q}^j$ is denoted as the set of structural fragments which have a distance to qe^j's native structure less than the distance threshold θ. A robust distance function should rank the structural fragments in $\mathcal{Q}^j$ for qe^j better than those non-native like structural fragments. Our objective here is to optimize the parameters of the distance function such that we have a distance function which ranks a qe^j's native-like structure better than which are not.

For notation simplicity, in the following formulations we assume $1 \leq i \leq p$ and $1 \leq j \leq q$ and the ILP is as follows:

$$\min \sum_{j=1}^{q} g_j \tag{3}$$

$$\mathcal{D}^{n,j} - \mathcal{D}^{i,j} \leq d_{n,i,j}(2+\epsilon) - \epsilon, \quad n \in \mathcal{Q}^j, i \notin \mathcal{Q}^j, \forall j \tag{4}$$

$$\sum_{1 \leq i \leq q, i \notin \mathcal{Q}^j} d_{n,i,j} \leq k + f_{n,j}(p-k), \qquad n \in \mathcal{Q}^j, \forall j \tag{5}$$

$$\sum_{n \in \mathcal{Q}^j} f_{n,j} \leq |Q^j| - 1 + g_j, \qquad \forall j \tag{6}$$

$$\sum_{l=1}^{\ell} w_l^j = 1, \qquad \forall j \tag{7}$$

$$d_{n,i,j}, f_{n,j}, g_j \in \{0,1\}, \qquad w_i^j \in [0,1] \tag{8}$$

The constant ϵ is created to tolerate errors. Ideally we want to have the parameter settings such that:

$$\mathcal{D}^{n,j} + \epsilon \leq \mathcal{D}^{i,j} \tag{9}$$

where se^n is a native-like structure for qe^j, and se^j is non native like structure for qe^j. Eq. 4 is used to achieve this goal. It is clear that $-2 \leq \mathcal{D}^{n,j} - \mathcal{D}^{i,j} \leq 2$. If $d_{n,i,j} = 0$, we rank the near native like structure se^n better than the non-native like structure se^i. Eq. 5 intends to score a native like structure se^n to be in the candidate list with size k. If $f_n, j = 0$, then the number of non native-like structural fragments for qe^j that scores higher than se^n (which is a native like structure for qe^j) is at most k. If $g_j = 0$, Eq. 6 ensures that at least one near native structure for sequence segment qe^j is in its candidate list. The objective function Eq. 3 is used to minimize the number of sequence segments whose candidate list with size k does not contain a near native structure. Eq. 7 is just to normalize the parameter distributions.

3.7 Distance Entries

The distance entries in this paper consist of four types of scores, which are mainly taken from Xu (2005).

3.7.1 Mutation Scores Mutation score is similar to the case of Rosetta, as shown in Eq. 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, $1 \leq i \leq \ell$ is defined to be:

$$S(aa,i) \times \log \frac{X(aa,i)}{S(aa,i)} \tag{10}$$

where we recall from Eq. 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 city block metric, dot product, and the one from Kim *et al.* (2003a), we find that Eq. 10 is slightly more stable.

3.7.2 Secondary Structure Score The secondary structure for a structural element is computed with DSSP (Kabsch and Sander, 1983). For the secondary structure of a sequence, we use PSIPRED (Jones, 1999). The program predict the confidences for a position to be α-helix, β-sheet and loop. Let the confidences predicted for each position i be α_i, β_i, and l_i. There are three cases when computing the secondary structure score at position i:

- 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.

3.7.3 Contact Capacity Score For each structural position $se[i]$, a contact number n_i is calculated. The contact capacity potential is due to consider hydrophobic contributions of free energy. Contact capacity is to measure the capacity that a residue has c contacts with any other residues in the template.

Given a protein template, 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: $S(aa,i) \times CC(n_i, aa)$.

3.7.4 Environmental Fitness Score The environments for each structural position are 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 9 states of structural environments and each structural position has one of the environment type. Denote the environment type at structural position i as E_i, and an amino acid aa fitness score for environment E as $F(E,aa)$, and then the fitness score between $se[i]$ and $qe[i]$ is calculated as: $S(aa,i) \times F(E_i, aa)$. For more details, we refer readers to Kim *et al.* (2003b).

3.8 Implementation

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

4 RESULTS

First we compare FragShaver's score function with Rosetta's fragment city block metric. Then we show that our program is promising for selecting structural candidates from a fragment library. Finally we show that our package can be applied to improve given distance function for fragment selection.

4.1 FragShaver's Score Function vs. City Block Metric

For the Rosetta's fragment selection method, we implemented it according to the specifications from Simons *et al.* (1997); Rohl *et al.* (2004).

Table 2 compares the City Block Metric (CBM) and FragShave's score (FSS) function. The fragment candidate list size is set to be 25, the number of templates used is 40 and the fragment length is 9.

Table 2. Position Coverage for CBM vs. FragShaver's Score Function

θ (Å)	α-Helix		β-Sheet		Loop		Overall	
	CBM	FSS	CBM	FSS	CBM	FSS	CBM	FSS
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. CBM is the City Block Metric. FSS is the FragShaver's Score function. 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 for Threshold Value as 1Å.

k	α-Helix		β-Sheet		Loop		Overall	
	CMB	FSS	CMB	FSS	CMB	FSS	CMB	FSS
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

Position coverage score(%) is displayed. CBM is the City Block Metric. FSS is the FragShaver's Score function. The first column is the fragment candidate list size. The fragment length is 9. The position coverage (%) is reported for the three cases. The threshold value is 1Å.

Table 4. Fragment Coverage and Local Fit Score for Threshold Value as 1Å.

k	Fragment Coverage(%)		Local Fit Score(Å)	
	CBM	FSS	CBM	FSS
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

CBM is the City Block Metric. FSS is the FragShaver's Score function. The first column is the fragment candidate list size. Column 2 and Column 3 are the fragment coverage scores for CBM and FSS, respectively. Column 4 and Column 5 are the local fit scores for CBM and FSS, respectively. The fragment length is 9. The threshold value is 1Å.

The table displays the position coverage. In Table 2, we can see that the improvement is small for α-helix. This is mainly because that the possible improvement gap is small. With the threshold value as 0.5Å, the position coverage increases from 10.0% to 37.6%, and from 26.6% to 38.7% for β-sheet and loop, respectively. With the threshold value as 1Å, the position coverage increases from 56.4% to 89.6%, and 55.5% to 78.1% for β-sheet and loop, respectively. For threshold 1.5Å, significant improvement is observed for β-sheet and loop as well. Overall, we can have a position coverage 88.2% and 96.7% for threshold value 1Å and 1.5Å , respectively, and the two values for CBM are 72.2 and 89.9.

In Table 3, we fix the threshold value as 1Å and we compare the results by varying the candidate list size. The position coverage is displayed. The improvement for β-sheet is more than 30% on average with the same candidate list size. The improvement for loop 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-sheet and loop, respectively, while reducing the fragment candidate size from 25 to 10 simultaneously. By using 5 as the candidate list size, FragShaver's performance is better than that of CBM with 40 as fragment candidate list size for β-sheet and loop. Also with using 15 as the candidate list size, FragShaver's performance is better than CBM with 40 as the candidate list size in all the cases.

In Table 4, we fix the threshold value as 1Å and we compare the results by varying the candidate list size. Table 4 shows the results of fragment coverage and local fit criteria. FragShaver with candidate list size as 10 has higher fragment coverage than the fragment coverage of CBM with candidate list size 40, and the scores are 43.3% and 40.8%, respectively.

For all these evaluation criteria, we can safely draw a conclusion that FragShaver 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. The FragShaver is stable and robust, and consistent improvement is observed.

4.2 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 subsection, we show that FragShaver 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, FragShaver 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 FragShaver'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.

4.3 Optimizing the Parameters

Our package can be employed to improve a designed score function. In this experiment, we assume the city block metric in Eq. 1 is designed as the distance function. The distance function contains $\ell \times 20$

Table 5. Customized Fragment Lists vs. Independent Fragment Libraries

	Fragment Coverage (%)		Local Fit Score (Å)	
L or k	KFL	FSS	KFL	FSS
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

KFL stands for Kolodny's fragment libraries. FSS is FragShaver's distance function. This first column is the fragment candidate list size for FragShaver, and is the library size for Kolodny's libraries. Fragment Coverage (%) of a threshold 0.5Åis shown for Kolodny's fragment libraries at column 2 and for FragShaver's distance function at column 3, respectively. Local fit score (Å) is shown for Kolodny's fragment libraries at column 4 and for FragShaver's distance function at column 5, respectively.

Table 6. Position Coverage of City Block Metric vs. Optimized City Block Metric

	α-Helix		β-Sheet		Loop		Overall	
θ (Å)	CBM	Opt	CBM	Opt	CBM	Opt	CBM	Opt
0.5	96.3	96.8	18.1	28.6	30.8	34.3	53.6	57.2
1	99.2	99.3	74.8	82.6	64.8	70.4	80.0	83.9
1.5	99.8	100	95.4	98.0	88.8	91.5	94.3	96.1
2	100	100	99.8	99.9	98.8	99.1	99.4	99.6
2.5	100	100	99.9	99.9	99.8	99.8	99.9	99.9
3	100	100	99.9	99.9	99.9	99.9	99.9	99.9
3.5	100	100	99.9	100	99.9	100	99.9	100
4	100	100	100	100	100	100	100	100

CBM is the City Block Metric. Opt is the optimized version of City Block Metric. The first column is the distant threshold (Å). The candidate list size is 20. The fragment length is 8.

items. We group every 20 items for each position together as a single score entry. Then we assign each entry a weight as a parameter, which makes the new distance function as:

$$DISTANCE = \sum_{i=1}^{\ell} w_l \sum_{aa}^{20} |S(aa,i) - X(aa,i)| \qquad (11)$$

We apply the FragShaver to CBM, and the result of the position coverage is shown in Table 6. The test is done using a candidate list size of 20, fragment length 8, and a total of 40 templates. For thresholds 0.5Å and 1Å, the coverage improved about 3-10% for β-sheet and loop. Improvements can be observed for all the cases. We conducted experiments with different lengths, candidate list sizes and evaluation criteria, and similar results were obtained.

Our package also allows faster prototyping of distance functions for fragments for specialists. One can simply test if distance entries are useful without worrying about the parameters, as the optimal combination of the parameters is guaranteed by the nature of the integer linear programming.

5 DISCUSSION

In our experiments we used the distance entries: mutations score, secondary structure score, contact capacity score, and environment fitness score, and then we use a linear combination of these distance entries. To improve performance, a natural idea is to use more distance entries. One way is to combine the scores from threading results, like the case in Zhang (2006). Moreover, all the scores currently used are assigned to single residues, which implies that we take the residues to be independent.

However, some residues are actually correlated, and it may be better to incorporate the correlation information into the system. One of the difficulties for the correlation information is the lack of training data, as there are 20×20 combinations for any two positions. A possible way to overcome this might be to create some pseudo-count techniques as in Blast.

Our work improves the accuracy of β-sheet and loop positions, and this gives us the probability to predict loop regions more accurately, as loops are considered to be the most variable parts in a protein structure. The program can 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.

The integer linear programming technique used in this work, can be applied to parameter training for other problems. The advantages for the integer linear programming model over support vector machine is that: integer linear program can find optimal combinations; and we can learn the insights more easily from the parameter values.

6 CONCLUSION

In this work, we developed a tool named FragShaver. It can build a customized structural candidate list for each sequence segment of a protein sequence and allow one to reconstruct the protein with high accuracy. The program uses integer linear programming to find the optimal combinations of the parameters, hence relieving the specialists of the troubles of parameters tuning, allowing one to prototype the ideas on distance entries fast for fragment selection.

By comparing our package to the Rosetta's fragment selection method, with the threshold as 1Å, and fragment length 9, the position coverage is improved from 56.4% to 79.1% for β-sheet, and from 55.5% to 67.9% for loop while reducing the candidate list size from 25 to 10 simultaneously. Our method is able to construct compact candidate lists for sequence segments and allows the protein structure to be reconstructed from these list accurately.

REFERENCES

Altschul, S., Madden, T., Schaffer, A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucl. Acids Res.*, **25**(17), 3389–3402.

Ashish V. Tendulkar, Anand A. Joshi, M. A. S. and Wangikar, P. P. (2004). Clustering of protein structural fragments reveals modular building block approach of nature. *Journal of Molecular Biology*, **338**, 611–629.

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

Bonneau, R., Tsai, J., Ruczinski, I., Chivian, D., Rohl, C., Strauss, C., and Baker, D. (2001). Rosetta in CASP4: progress in ab initio protein structure prediction. *Proteins*, **37**(S5), 119–126.

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. *Proceedings of the National Academy of Sciences of the United States of America*, **91**(10), 4436–4440.

Bradley, P., Chivian, D., Meiler, J., Misura, K. M., Rohl, C. A., Schief, W. R., Wedemeyer, W. J., Schueler-Furman, O., Murphy, P., Schonbrun, J., Strauss, C. E., and Baker, D. (2003). Rosetta predictions in CASP5: Successes, failures, and prospects for complete automation. *Proteins: Structure, Function, and Genetics*, **53**(S6), 457–468.

Bystroff, C. and Baker, D. (1998). Prediction of local structure in proteins using a library of sequence-structure motifs. *Journal of Molecular Biology*, **281**(3), 565–577.

Camproux, A., Tuffery, P., Chevrolat, J., Boisvieux, J., and Hazout, S. (1999). Hidden Markov model approach for identifying the modular framework of the protein backbone. *Protein Eng.*, **12**(12), 1063–1073.

Camproux, A. C., Gautier, R., and Tuffery, P. (2004). A hidden markov model derived structural alphabet for proteins. *Journal of Molecular Biology*, **339**, 591–605.

Chivian, D., D.E., K., Malmstrom, L., Schonbrun, J., Rohl, C., and Baker, D. (2005). Rosetta predictions in CASP5: Successes, failures, and prospects for complete automation. *Proteins: Structure, Function, and Genetics*, **61**(S7), 157–166.

Claessens, M., van Cutsem, E., Lasters, I., and Wodak, S. (1989). Modelling the polypeptide backbone with 'spare parts' from known protein structures. *Protein Eng.*, **2**(5), 335–345.

de Brevern, A. and Hazout, S. (2000). Hybrid protein model (hpm): A method to compact protein 3d-structure information and physicochemical properties. *spire*, **00**, 49.

de Brevern, A., Etchebest, C., and Hazout, S. (2000). Bayesian probabilistic approach for predicting backbone structures in terms of protein blocks. *Proteins: Structure, Function, and Genetics*, **41**, 271–287.

de Brevern, A. G., Valadie, H., Hazout, S., and Etchebest, C. (2002). Extension of a local backbone description using a structural alphabet: A new approach to the sequence-structure relationship. *Protein Science*, **11**(12), 2871–2886.

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

Etchebest, C., Benros, C., Hazout, S., and de Brevern, A. G. (2005). A structural alphabet for local protein structures: Improved prediction methods. *Proteins: Structure, Function, and Bioinformatics*, **59**(4), 810–827.

Fetrow, J. S., Palumbo, M. J., and Berg, G. (1997). Patterns, structures, and amino acid frequencies in structural building blocks, a protein secondary structure classification scheme. *Proteins: Structure, Function, and Genetics*, **27**(2), 249–271.

Fidelis, K., Stern, P. S., Bacon, D., and Moult, J. (1994). Comparison of systematic search and database methods for constructing segments of protein structure. *Protein Eng.*, **7**(8), 953–960.

Han, K. F., Bystroff, C., and Baker, D. (1997). Three-dimensional structures and contexts associated with recurrent amino acid sequence patterns. *Protein Sci*, **6**(7), 1587–1590.

Haspel, N., Tsai, C.-J., Wolfson, H., and Nussinov, R. (2003). Reducing the computational complexity of protein folding via fragment folding and assembly. *Protein Sci*, **12**(6), 1177–1187.

Holmes, J. B. and Tsai, J. (2004). Some fundamental aspects of building protein structures from fragment libraries. *Protein Sci*, **13**(6), 1636–1650.

Hunter, C. G. and Subramaniam, S. (2003). Protein fragment clustering and canonical local shapes. *Proteins: Structure, Function, and Genetics*, **50**(4), 580–588.

Inbar, Y., Benyamini, H., Nussinov, R., and Wolfson, H. J. (2003). Protein structure prediction via combinatorial assembly of sub-structural units. *Bioinformatics*, **19**(S1), 158–168.

J. Meller and R. Elber (2001). Linear programming optimization and a double statistical filter for protein threading protocols. *Proteins*, **45**(3), 241–261.

Jones, D. T. (1999). Protein secondary structure prediction based on position-specific scoring matrices. *Journal Molecular Biology*, **292**(2), 195–202.

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

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

Kim, D., Xu, D., Guo, J., Ellrott, K., and Xu, Y. (2003a). Prospect ii: Protein structure prediction method for genome-scale applications. *Protein Engineering*, **16**(9), 641–650.

Kim, D., Xu, D., tao Guo, J., Ellrott, K., and Xu, Y. (2003b). PROSPECT II: protein structure prediction program for genome-scale applications. *Protein Eng.*, **16**(9), 641–650.

Kolodny, R., Koehl, P., Guibas, L., and Levitt, M. (2002). Small libraries of protein fragments model native protein structures accurately. *JMB*, **323**, 297–307.

Kuhlman, B., Dantas, G., Ireton, G. C., Varani, G., Stoddard, B. L., and Baker, D. (2003). Design of a Novel Globular Protein Fold with Atomic-Level Accuracy. *Science*, **302**(5649), 1364–1368.

Lee, J., Kim, S.-Y., and Lee, J. (2005). Protein structure prediction based on fragment assembly and parameter optimization, Biophysical Chemistry. *Biophysical Chemistry*, **115**(2-3), 209–214.

Levitt, M. (1992). Accurate modeling of protein conformation by automatic segment matching. *Journal of Molecular Biology*, **226**(2), 507–533.

Micheletti, C., Seno, F., and Maritan, A. (2000). Recurrent oligomers in proteins - an optimal scheme reconciling accurate and concise backbone representations in automated folding and design studies. *PROTEINS*, **40**, 662.

Moult, J., Hubbard, T., Fidelis, K., and Pedersen, J. (1999). Critical assessment of methods of protein structure prediction (casp):round iii. *Proteins: Structure, Function and Genetics*, **37**, 2–6.

Moult, J., Fidelis, K., Zemla, A., and Hubbard, T. (2001). Critical assessment of methods of protein structure prediction (casp):round iv. *Proteins: Structure, Function and Genetics*, **45**, 2–7.

Moult, J., Fidelis, K., Zemla, A., and Hubbard, T. (2003). Critical assessment of methods of protein structure prediction (casp):round v. *Proteins: Structure, Function and Genetics*, **53**, 334–339.

Moult, J., Fidelis, K., Rost, B., Hubbard, T., and Tramontano, A. (2005). Critical assessment of methods of protein structure prediction (casp):round 6. *Proteins: Structure, Function and Genetics*, **61**, 3–7.

Pauling, L. and Corey, R. B. (1951). The Pleated Sheet, a New Layer Configuration of Polypeptide Chains. *PNAS*, **37**(5), 251–256.

Pauling, L., Corey, R. B., and Branson, H. R. (1951). The Structure of Proteins: Two Hydrogen-Bonded Helical Configurations of the Polypeptide Chain. *PNAS*, **37**(4), 205–211.

Prestrelski, S. J., Jr., A. L. W., and Liebman, M. N. (1992). Generation of a substructure library for the description and classification of protein secondary structure. i. overview of the methods and results. *Proteins: Structure, Function, and Genetics*, **14**, 430–439.

Rohl, C. A., Strauss, C. E., Misura, K. M., and Baker, D. (2004). Protein structure prediction using rosetta. *Methods Enzymol*, **383**, 66–93.

Rooman, M. J., Rodriguez, J., and Wodak, S. J. (1990). Automatic definition of recurrent local structure motifs in proteins. *Journal of Molecular Biology*, **213**(2), 327–336.

Schuchhardt, J., Schneider, G., Reichelt, J., Schomburg, D., and Wrede, P. (1996). Local structural motifs of protein backbones are classified by self-organizing neural networks. *Protein Eng.*, **9**(10), 833–842.

Simon, I., Glasser, L., and Scheraga, H. (1991). Calculation of Protein Conformation as an Assembly of Stable Overlapping Segments: Application to Bovine Pancreatic Trypsin Inhibitor. *PNAS*, **88**(9), 3661–3665.

Simons, K., Kooperberg, C., Huang, E., and Baker, D. (1997). Assembly of Protein Tertiary Structures from Fragments with Similar Local Sequences using Simulated Annealing and Bayesian Scoring Functions. *Journal of Molecular Biology*, **268**.

Simons, K. T., Bonneau, R., Ruczinski, I., and Baker, D. (1999). Ab initio protein structure prediction of CASP III targets using ROSETTA. *Proteins*, **37**(S3), 171–176.

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

Unger, R. and Sussman, J. L. (1993). The importance of short structural motifs in protein structure analysis. *Journal of Computer-Aided Molecular Design*, **7**(4), 457–472.

Unger, R., Harel, D., Wherland, S., and Sussman, J. L. (1989). A 3D building blocks approach to analyzing and predicting structure of proteins. *Proteins: Structure, Function, and Genetics*, **5**(4), 355–373.

Wang, G. and Roland L., J. D. (2003). PISCES: a protein sequence culling server. *Bioinformatics*, **19**(12), 1589–1591.

Wendoloski, J. and Salemme, F. (1992). Probit: a statistical approach to modeling proteins from partial coordinate data using substructure libraries. *J. Mol. Graph.*, **10**(2), 124–126.

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

Zhang, Y. (2006). Server prediction by iterative tasser simulations in casp7. In *CASP 7 Meeting Abstracts*, pages 123–124.

Zhang, Y., Arakaki, A., and Skolnick, J. (2005). TASSER: An automated method for the prediction of protein tertiary structures in CASP6. *Proteins*, **61**(S7), 91–98.

Zhou, H. and Zhou, Y. (2005). PARKS 2 and SP^3 servers in CASP6. *Proteins: Structure, Function, and Bioinformatics*, **61**(S7), 152–156.