A Comparison of Suffix Tree based Indexing and Search Techniques for Querying Protein Structures

Sanjay Kulhari
University of California, Riverside
skulhari@cs.ucr.edu

March 2010

Abstract

Biological research comes across different protein structures inside a cell which may be required to map to known proteins to quickly determine their functionality. Efficient techniques for searching a protein structure in a database containing all the known proteins are needed to classify the protein and predict its function. Comparing the structure of unknown protein individually with every protein in the database can be highly inefficient. Database indexing methods are therefore best suited for matching an unknown protein structure with the existing set of proteins. Various indexing techniques have been proposed till date that uses various features of protein structure. Indexing techniques based on suffix tree data structure are of prime importance as they provide efficient querying algorithms. In this report we are going to do a comparison of three such techniques that extracts features from protein structures, encode them to create a sequence, build a suffix tree index and provide algorithm to query the index for unknown protein.

1 Introduction

Computational methods are widely used to analyze biological data. Identifying proteins is one of such task that requires comparing a protein structure against a database that contains structures of known proteins. Proteins are chain of amino acids that are arranged in three dimensions. The distance between carbon atoms, angles formed between different atoms in the amino acid forms the three dimensional structure. This three dimensional structure then determines the functionality of the protein that the main purpose of protein classification. Simply comparing the sequence of amino acids in a protein to another protein may not help to classify it and determine its functionality. It may also happen that the two proteins having similar sequence have totally different functionality because they have different three dimensional structure.

Indexing techniques for proteins need to take into account the structure which is also called as feature of the protein. Feature of a protein includes the geometry that cannot be represented as a single entity, so before indexing, this geometry has to be converted to concrete data type like a string or a numeral that can be indexed. In this report we are going to see three approaches that

use structural information to build suffix tree for proteins. The three approaches are called PSIST (Protein Structure Indexing using Suffix Trees) [1], Geometric Suffix Tree (GST) [2] and Protein Similarity Algorithm (PROSIMA)[3]. Since these approaches are different in extracting the features, the information stored in the index, query techniques and efficiency will be different. Accuracy for the three techniques will be different as it depends on what information is used to do build the index. In further sections, we are going to do a comparison based on feature extraction, suffix tree creation, querying, accuracy and time complexities.

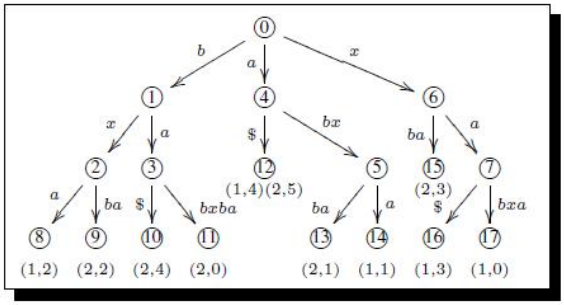
2 Preparing data

As previously mentioned the general methodology is quite similar and all the techniques converts structural information into a suffix tree, but the information stored in the tree can be substantially different. PROSIMA and PSIST are considered efficient to GST, but less accurate. GST is based on calculating Root Mean Square Deviation (RMSD) and it constructs a suffix tree where the edges represent 3-D substructures as compared to sequence of strings in the other two approaches.In this section we are going to see how the features are extracted and data is prepared for construction of suffix trees. From the chain of carbon atoms PSIST uses a sliding window to extract local feature vector that records the Euclidean distance between the alpha carbon atoms and the angle formed by planes that contain those carbon atoms. Since the relationship between two amino acids can be described by its dihedral angles, as compared to PSIST, PROSIMA just encodes the three dihedral angles into a string and then uses a dictionary to encode them again into a positive integer. We see that PSIST uses sliding window, distance between carbon atoms, and angles between the planes of carbon atoms, in contrast PROSIMA just uses the three angles and consider just the two adjacent carbon atoms in the chain. Thus we can say the information contained in PSIST is more than the information contained in index created by PROSIMA. GST on the other hand, based on the 3D coordinates of the carbon atoms calculates RMSD and uses that information for tree creation. GST considers all the carbon atoms and thus is a global approach as compared to PSIST and PROSIMA that uses just the local information from a number of carbon atoms in the amino acid chain. Since preparing the data has its own complexity, let us now look in more detail how these techniques differ in the way data is prepared.

In PSIST, Euclidean distance between two alpha-carbon atoms is directly calculated using 3D coordinates and the angle between the planes is calculated by finding the angle between the normal of the two planes. Since this techniques uses a sliding window, the distance and angle is calculated between the alpha-carbon atom of consideration and the carbon atoms of other amino acids present in the window. If the window size is w, the dimension of each feature vector is $2(w-1)$ because we are considering the distance as well as the angle. After getting these values, they are normalized and binned to an integer in the range [0, b-1]. The sequence of normalized values is called structure-feature sequence. Since each normalized value is in range [0, b-1], the structure-feature sequences are over an alphabet of size $b^{2(w-1)}$. Each normalized feature vector is assigned a unique identifier as a label and thus structure-feature sequence for a protein is nothing but a sequence of labels for which a suffix tree is constructed.

Table 1. Examples of normalized feature vectors for $w = 3$ and $b = 10$

	Feature vector			
	d	$\cos\theta$	d	$\cos\theta$
original	3.55	0.29	5.4	−0.23
normalized	4	6	6	3
original	4.04	0.11	5.75	−0.25
normalized	5	5	7	3
original	3.60	0.45	5.29	0.21
normalized	4	7	6	6



GST for sequences $S_1 = xabxa$ and $S_2 = babxba$

Figure 1: Structure feature vectors and it's generalized suffix tree

PROSIMA uses the dihedral angle information (ϕ, Ψ and ω) from the adjacent amino acids. It gets all the protein files from PDB database and encodes the amino acid chain into a sequence of integers. Since ϕ and Ψ takes values from -180 to 180, an interval of 10 is created and the range of possible values is broken into 35 parts. If the angle is say -21 then it can be mapped to nearest discrete value of 20 and normalized to a value of 02. This way the two angles can be encoded as strings. ω on the other hand can take values of either 0 or 180, so it is encodes as A or B. As an example if we have a protein with backbone consisting of 6 amino acid residues MVLSEG, the resulting encoded sequence can be A3202, A2401, A2603, B2401, A2422. This way each protein is encoded as sequence of words, which can be converted to sequence of integers by building a dictionary on these words. Thus we get a protein as sequence of integers for which a suffix tree can be constructed.

Geometric Suffix Tree is considerably different from the other two techniques as it does not index just some important features of structures. From the set of available proteins, it calculates the pairwise root mean square deviation (RMSD) between the corresponding atoms of the protein structure. Since this is like a holistic approach, it is going to be more accurate but considerably slow. Also unlike PROSIMA, it works for only a small set of proteins, otherwise it can be highly inefficient. Depending on the RMSD values and minimum RMSD limit between two structures, addition of nodes in the suffix tree is determined. In the next section we are going to see how the prepared data in these techniques is used to construct the suffix tree.

3 Indexing

Structure-feature sequences for each protein structure obtained for PSIST in data preparation phase is a sequence of labels. Given such sequences for all the proteins, a generalized suffix tree is constructed.

As an example, in Figure 1, there are three feature vectors with a window size of three. We see that each feature vector is a sequence of normalized values. Vector 1 is 4-6-6-3; similarly vector 2

is 5-5-7-3 and vector 3 is 4-7-6-6. The label assigned to the three vectors is a, b and x respectively. If the two sequences are $S1 = xabxa$ and $S2 = babxba$, a generalized suffix tree is constructed as shown in figure 1.

In case of PROSIMA, protein structure was encoded as sequence of positive integers. As PROSIMA builds the suffix tree for all the proteins in the PDB database, an encoding of all the protein structures is done and a generalized suffix tree is created. As a simple example, consider three protein structures with a chain of six amino acids encoded as 01213, 23162, and 12423 respectively. These sequences of integers are used to construct a generalized suffix tree. In this technique as we said, generalized suffix tree is created using sequences of all the proteins in the database.

Suffix tree construction for Geometric suffix tree requires all the carbon atoms to be considered and RMSD is calculated for each protein structure against all available protein structures that are to be indexed. Suffix substructure is a sequence of suffixes in the chain of carbon atoms that represents the 3-D coordinates of the carbon atoms in the sequence. During suffix tree creation, suffix substructures are inserted into the suffix tree as compared to sequence of labels and numbers in PSIST and PROSIMA respectively.

In the next section we are going to see how the information stored in the suffix tree index is used to compare the structure of query protein with the encoded structures of protein in the database.

4 Querying

In PSIST, query's feature vectors are extracted in the same way and converted to structure-feature sequence as it was done while indexing proteins. The complete querying process is divided into three phases - searching, ranking and post processing. Search here is not an exact match in contrast to usual suffix tree maximal unique matches. It takes into consideration two parameters that are ϵ, which is the maximum possible distance between matched strings and the minimum length l of the maximal match. Since we are interested in finding matches that are with a small distance, every subsequence of the query can be traced in the suffix tree, but this may cause common prefix of two subsequences to be searched twice. Thus to avoid the unnecessary comparisons, a suffix tree is built on the suffixes of the query sequence. A set of all maximal matches is created and if a maximal match is found it is added to it. The next step is to chain the matches as to identify the protein sequences with high matching scores so that we can narrow down to smaller number of proteins to perform alignment with the query. Chaining the maximal matches is based on greedy algorithm that chooses the match with the highest score and all the overlapping matches are removed and the process is repeated for next match with highest score. Finally proteins sequences with top scores are selected and aligned with the query using Smith-Waterman algorithm. The protein with the highest score is reported to the user.

PROSIMA evaluates the similarity in proteins using standard information retrieval way of building a vector model and a similarity matrix. To build the vector model it needs to query the suffix tree index to obtain maximal phrase cluster which is nothing but the longest common substructure.

The protein vector model is analogous to document model represented by terms. In the protein model, the document is the encoded chains of amino acids and the terms are the maximal phrase clusters. A protein is represented by the maximal phrase cluster in which it is contained. Similar to the weight of terms in document model, weights of phrase clusters is calculated. Thus the vector model for proteins is generated and similarity matrix is created by using cosine similarity measure. The similarity matrix can report the most similar protein to a protein present in the vector model by finding the protein with highest similarity score in its row. This approach indexes all the proteins in the database and finds the similarity matrix for them. This technique can also be employed to find the most similar protein to the protein in the query by indexing the query protein along with the database proteins.

In case of GST, query takes the form Q[1..m], where Q is the sequence of carbon atoms with its 3-D coordinates. We know that instead of extracting the feature and converting them to a sequence of labels, edge in a GST represents the substructure itself. Given a query Q, GST can return all maximal substructures whose RMSD is within some bound $'d'$. Representative structure is defined as prefix substructure of the node structure in the geometric suffix tree. To identify the maximal substructures for a given query, all the maximal representative substructures are found whose RMSD to query Q is within $\sqrt{b/m} + d$ where b is the bound used for constructing the geometric suffix tree. All the suffixes to leaves that are descendent to the edges of representative substructure will be reported to the user

5 Accuracy, Space and Time complexities

In this section we are going to compare complexities and accuracy of the three approaches. Let the number of proteins to be indexed is N and the sum of the lengths of the protein chains is L and the length of query protein is M. The techniques that we have described in the report make use of suffix tree to index the available set of proteins. PROSIMA works on the entire PDB database, and extracts the features using the three dihedral angles and the adjacent carbon atoms. Data preparation phase requires converting protein structures in the database to sequences, complexity of which is of the order of sum of the lengths $O(L)$ of the protein chains. Suffix tree creation algorithm used in PROSIMA is quadratic and is $O(L^2)$. Querying the suffix tree for longest common substructure is $O(L)$, complexity of building the similarity matrix is $O(N^2)$. Therefore the total complexity is $O(L^2 + N^2)$. This complexity is for finding similarity of every protein in the database to every other protein. Thus if we use this technique for finding similarity of just the query protein, complexity can be reduced to $O(N)$. Also an efficient implementation of suffix tree construction can reduce the construction cost to $O(L)$ i.e linear in the sum of the lengths of the protein chains. The space complexity for PROSIMA is $O(L)$ for index creation, but the over all space complexity for building the similarity matrix is $O(N^2)$. It is least accurate compared to the other two techniques.

Data preparation phase in PSIST is also of the order L if each protein chain is processed with a window size 2. It will be $O(wL)$ for a window size w. Since the window size w is not expected to be a big number, complexity can be assumed to be $O(L)$. PSIST uses linear time suffix tree construction

algorithm. Since each protein will be represented by a sequence of labels whose count is same as the length of the protein chain, the suffix tree creation algorithm is linear to the sum of lengths of all proteins. Therefore creation of sufix tree is $O(L)$. Complexity for querying the suffix tree for maximal matches is $O(M + k)$, where k is the number of matches. These matches are ranked and using greedy algorithm it takes $O(k)$ time to delect top matches. Assuming that a very small number of proteins say c are selected as top matches, post processing step of using dynamic programming method to find the alignment will be $O(c * (l_i * M))$, where l_i is the length of one of the protein selected as top match. Assuming the size of protein chain to be of the same order as size of the query, post processing step can be viewed as $O(M^2)$. Suffix tree construction algorithm for PSIST has space complexity of $O(L)$. The experimental results show the accuracy of 97.8%.

Data preparation phase in geometric suffix tree can be viewed as calculation of RMSD between the protein structures. RMSD calculation and insertion of a protein structure in the suffix trie requires time proportional to the sum of size of the structures and the number of proteins. Thus this step has time complexity of $O(L+N^2)$. Construction of suffix tree is $O(L^2+LN^2)$. Query execution to find maximal substructures takes time proportional to sum of the lengths of the protein chains. So the complexity of finding the mathed result is $O(L)$. Also the space complexity for GST is $O(L)$. GST is the most accurate among these indexing techniques.

6 Conclusion

In this report we compared three indexing techniques that are based on suffix trees. GST is inefficient compared to other two techniques but is more accurate as it constructs the index based on pair wise comparison. PROSIMA uses minimal feature information thus is expected to be less accurate. PSIST is the most efficient among these and is also more accurate than PROSIMA as it takes a number of residues in the window to create the feature vector. The decision of selecting a technique depends on required accuracy and efficiency. We have not studied other index based techniques that do not use suffix trees, so these techniques are not compared to them in this report. Taking into consideration just these three techniques, if the unknown protein is compared to a reasonably small subset of proteins, GST can be used otherwise for querying against the complete PDB database PSIST should be the preferred technique.

References

[1] Feng Gao,Mohammed J. Zaki PSIST: Indexing Protein Structures Using Suffix Trees. In *IEEE Computational Systems Bioinformatics Conference*, August 2005.

[2] Tetsuo Shibuya Geometric Suffix Tree: A New Index Structure for Protein 3-D Structures. In *Annual Symposium on Combinatorial Pattern Matching*, July 2006.

[3] Tomas Novosad, Vaclav Snasel, Ajith Abraham, Jack Y Yang PROSIMA: Protein Similarity Algorithm. In *Nature and Biologically Inspired Computing*, December 2009.