Prefix-Shuffled Geometric Suffix Tree

Tetsuo Shibuya

Human Genome Center, Institute of Medical Science, University of Tokyo
4-6-1 Shirokanedai, Minato-ku, Tokyo 108-8639, Japan
tshibuya@hgc.jp

Abstract. Protein structure analysis is one of the most important research issues in the post-genomic era, and faster and more accurate index data structures for such 3-D structures are highly desired for research on proteins. The geometric suffix tree is a very sophisticated index structure that enables fast and accurate search on protein 3-D structures. By using it, we can search from 3-D structure databases for all the substructures whose RMSDs (root mean square deviations) to a given query 3-D structure are not larger than a given bound. In this paper, we propose a new data structure based on the geometric suffix tree whose query performance is much better than the original geometric suffix tree. We call the modified data structure the prefix-shuffled geometric suffix tree (or PSGST for short). According to our experiments, the PSGST outperforms the geometric suffix tree in most cases. The PSGST shows its best performance when the database does not have many substructures similar to the query. The query is sometimes 100 times faster than the original geometric suffix trees in such cases.

1 Introduction

Protein 3-D structure analysis is one of the most important post-genomic research topics in molecular biology. Recently, more and more protein structures are solved by state-of-the-art technologies such as NMR (nuclear magnetic resonance), and the size of the protein 3-D structure database increases larger and larger. Now, there are more than 40,000 entries in the PDB database [2] and it is still increasing. The protein structures are said to have similar functions if their 3-D structures are similar. Thus, to analyze the functions of a protein whose structure is newly determined, it is very important to search for similar (sub)structures from the growing database. There are many comparison algorithms for protein structures [5], and the results could be very accurate, but it will require enormous amount of time to apply them against the very large databases. Hence, indexing techniques for protein structure databases are highly desired to avoid the large computation time.

The similarity of two protein structures is often measured by the RMSD (root mean square deviation) [1,4,11]. The geometric suffix tree [12] is an indexing data structure that enables efficient search from a 3-D structure database for all the substructures whose RMSDs to a given query are not larger than some given bound. It also has many potential applications, such as 3-D motif finding and

N. Ziviani and R. Baeza-Yates (Eds.): SPIRE 2007, LNCS 4726, pp. 300–309, 2007.

functional prediction. The geometric suffix tree is based on the famous suffix trees for alphabet strings [6,8,10,13,14], but it deals with 3-D coordinates instead of alphabet characters. In this paper, we propose a new data structure based on the geometric suffix tree, which we call the prefix-shuffled geometric suffix tree, or PSGST for short. It improves the query performance of the geometric suffix tree by changing the order of atoms in each substructure. We will demonstrate the PSGSTs' performance through experiments.

This paper is organized as follows. In section 2, we explain the preliminaries. In section 3, we explain a new notion called the 'prefix-shuffled structure' that would help us to improve the query performance of the geometric suffix trees. Then, in section 4, we explain the newly proposed data structure, the prefix-shuffled geometric suffix tree. In section 5, we demonstrate the performance of it through experiments. Finally in section 6, we conclude our results and discuss future work.

2 Preliminaries

2.1 RMSD: The Root Mean Square Deviation

A protein is a chain of amino acids. Each amino acid has one unique carbon atom named C_α, and the set of all the C_α atoms in a protein is called the backbone of the protein. The backbone is topologically linear, but it forms a geometrically very complex structure in the 3-D space. Most previous work on protein 3-D structures deals with the coordinates of the backbone atoms. Thus, we also consider the coordinates of the backbone atoms as the target to index. The most popular and basic measure to determine geometric similarity between two sets of points in 3-D, like the positions of backbone atoms, is the RMSD (root mean square deviation) [1,4,11].

Before defining the RMSD, let us define the measures that we call the MSSD (minimum sum squared distance) and the RSSD (Root Sum Square Distance). Let the two sets of points (*i.e.*, structures) to be compared be $P=\{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n\}$ and $Q = \{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n\}$, where $\boldsymbol{p}_i$ and $\boldsymbol{q}_j$ are 3-D coordinates. To compute the MSSD/RSSD/RMSD between two sets of 3-D coordinates, we must know which atom in one structure corresponds to which atom in the other. Here we consider $\boldsymbol{p}_i$ corresponds to $\boldsymbol{q}_i$ for all i. Let $mssd(P, Q)$ be the minimum value of $\sum_{i=1}^{n} \|\boldsymbol{p}_i - (R \cdot \boldsymbol{q}_i + \boldsymbol{v})\|^2$ over all the possible rotation matrices R and translation vectors $\boldsymbol{v}$, where $\|\cdot\|$ denotes the norm. Let $\hat{R}(P, Q)$ and $\hat{\boldsymbol{v}}(P, Q)$ be the rotation matrix and the translation vector that satisfies $\sum_{i=1}^{n} \|\boldsymbol{p}_i - (\hat{R}(P, Q) \cdot \boldsymbol{q}_i + \hat{\boldsymbol{v}}(P, Q))\|^2 = mssd(P, Q)$. Then the RSSD is defined as the squared root of it: $rssd(P, Q) = \sqrt{mssd(P, Q)}$, and the RMSD is finally defined as $rssd(P, Q)/\sqrt{n}$.

It is known that $\hat{\boldsymbol{v}}(P, Q) = \sum_{i=1}^{n} (\boldsymbol{p}_i - \hat{R}(P, Q) \cdot \boldsymbol{q}_i)/n$. It means that the centroids of the two point sets must be translated to the same point by $\hat{\boldsymbol{v}}(P, Q)$. Hence, if both of the point sets are translated so that their centroids are located at the origin of the coordinates, the RMSD problem is reduced to a problem of finding R that minimizes $f(R) = \sum_{i=1}^{n} \|\boldsymbol{p}_i - R \cdot \boldsymbol{q}_i\|^2$. We can solve this problem

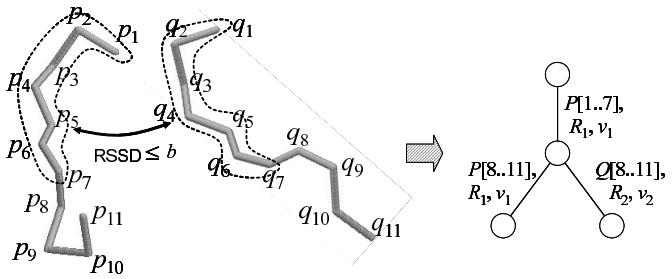

Fig. 1. A geometric trie for two protein 3-D structures. A node is constructed for $P[1..7]$ and $Q[1..7]$, as the RSSD between $P[1..7]$ and $Q[1..7]$ is smaller than the threshold b_{RSSD}. The combined edge is represented by arbitrary one of the two substructures — $P[1..7]$ is chosen in this example.

in linear time by using the singular value decomposition (SVD) [1] as follows. Let $H = \sum_{i=1}^{n} \boldsymbol{p}_i \cdot \boldsymbol{q}_i^t$, where $\boldsymbol{v}^t$ means the transpose of vector $\boldsymbol{v}$. Then $f(R)$ can be described as $\sum_{i=1}^{n} (\boldsymbol{p}_i^t \boldsymbol{p}_i + \boldsymbol{q}_i^t \boldsymbol{q}_i) - trace(R \cdot H)$, and $trace(RH)$ is maximized when $R = VU^T$, where $U\Lambda V$ is the SVD of H, and U^T denotes the transpose of matrix U. The SVD of H can be done in constant time as H is a fixed-size 3×3 matrix (see [7] for SVD algorithms). Hence the optimal rotation matrix can be obtained in constant time from H. In this way, we can compute the RMSD in $O(n)$ time. Note that there are rare degenerate cases where $det(VU^T) = -1$, which means that VU^T is a reflection matrix. We ignore the degenerate cases in this paper.

According to [12], the RMSD value, the optimal rotation matrix $\hat{R}(P, Q)$, and the optimal translation vector $\hat{\boldsymbol{v}}(P, Q)$ can be computed incrementally by keeping some additional values for computation, *i.e.*, we can compute $\hat{R}(P_i, Q_i)$ and $\hat{\boldsymbol{v}}(P_i, Q_i)$ for $P_i = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_i\}$ and $Q_i = \{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_i\}$ in $O(1)$ time after the computation of $\hat{R}(P_{i-1}, Q_{i-1})$ and $\hat{\boldsymbol{v}}(P_{i-1}, Q_{i-1})$ for P_{i-1}=$\{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{i-1}\}$ and $Q_{i-1} = \{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_{i-1}\}$, for any i (see [12] for more details).

2.2 Geometric Suffix Trees

The suffix tree of a string is the compacted trie of all its suffixes. Likewise, the geometric suffix tree [12] is based on a data structure called the *geometric trie*, which is defined as follows.

Consider a set of n 3-D structures $\mathbf{W} = \{W_1, W_2, \ldots, W_n\}$, and let ℓ_i be the length of W_i. Let $\boldsymbol{w}_j^{(i)}$ denote the coordinates of the j-th atom of W_i. Let $W_i[j..k]$ denote $\{\boldsymbol{w}_j^{(i)}, \boldsymbol{w}_{j+1}^{(i)}, \ldots, \boldsymbol{w}_k^{(i)}\}$, which means a structure formed by the $(k-j+1)$ atoms from the j-th atom to the k-th atom in W_i. We call it a substructure of W_i. Furthermore, we call $W_i[1..j]$ $(1 \leq j \leq \ell_i)$ a prefix substructure of W_i. Conversely, $W_i[j..\ell_i]$ is called a suffix substructure. Then, the geometric trie for $\mathbf{W}$ is a rooted tree data structure that has the following features (Figure 1):

1. All the internal nodes (nodes other than the root and the leaves) have more than one child.
2. The tree has n leaves, each of which corresponds to one protein structure in $\mathbf{W}$, and no two leaves correspond to the same structure. Let $leaf(i)$ denote the leaf that corresponds to W_i.
3. All the edges e except for some of edges that end at leaves correspond to a substructure $P(e) = W_i[j..k]$, and they have information of some 3-D rotation matrix $R(e)$ and some 3-D translation vector $\boldsymbol{v}(e)$ for each.
4. Let $S(e)$ be $P(e)$ rotated by $R(e)$ and translated by $\boldsymbol{v}(e)$, which is called the 'edge structure' of e. For a node x in the tree, let $S(x)$ be a structure that is constructed by concatenating all the edge structures of the edges on the path from the root to x, which we call the 'node structure' of x. For any leaf $v = leaf(i)$ and its node structure $S(v)$, the RSSD between any prefix substructure of $S(v)$ and the prefix substructure of W_i (of the same length) must not be larger than some given fixed bound b_{RSSD}.
5. For an edge $e = (v, w)$ with some corresponding substructure $P(e)$, the 'branching structure' $str(e)$ is defined as a structure that is obtained by adding the coordinates of the first atom of $S(e)$ (*i.e.*, $S(e)[1]$) after $S(v)$. For any internal node v with more than one outgoing edge with corresponding substructures, the RSSD between $str(e_1)$ and $str(e_2)$ must be larger than b_{RSSD}, where e_1 and e_2 are arbitrary two of the edges.

Then the *geometric suffix tree* of a structure $P = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n\}$ is defined as the geometric trie of all the suffix substructures of P. The geometric suffix tree can be stored in $O(n)$ memory, though there are $O(n^2)$ substructures in the target structure. It can be built in $O(n^2)$ time by just adding suffix substructures into the tree one by one, using the incremental RMSD computation technique. The geometric suffix tree can be easily extended to deal with all the suffix substructures of a set of structures, like the generalized suffix trees for ordinary alphabet strings [8].

A prefix substructure of a node structure is called a 'representative structure'. To search for a substructure similar (*i.e.*, RMSD is within some bound b_{RMSD}) to a query $Q[1..m]$ using the geometric suffix tree, we first search for all the representative structures of length m whose RMSD to Q is within $b_{RMSD}+(b_{RSSD}/\sqrt{m})$. There always exist (one or more) original substructures that correspond to each representative structure. Finally, if the RMSDs between the query and the enumerated original substructures are actually within b_{RMSD}, we output them as the answers.

3 Prefix-Shuffled Structures

When we search for similar substructures from the geometric suffix trees, we incrementally compare RSSDs between the prefix substructures of the query structure and representative structures. In Figure 2, the line noted as 'Normal' shows the RSSDs of prefix substructures (of various lengths) of two very different

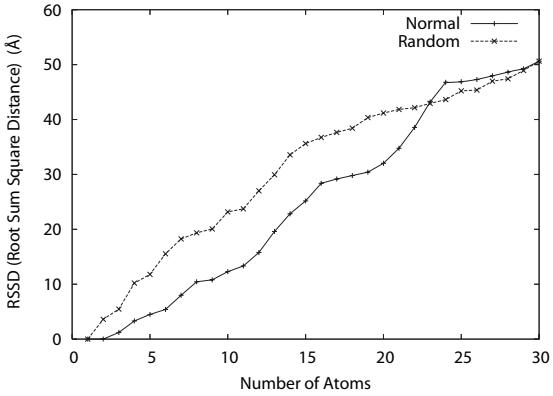

Fig. 2. Prefix RSSDs and Shuffled Prefix RSSDs. The RSSD goes up faster if the order of the atoms are shuffled.

proteins (a myoglobin and a rhodopsin taken from the set of structures used in section 5). In this example, the RMSD between two prefix substructures of length 30 is 9.40Å (*i.e.*, RSSD is 50.62Å), which means that the two structures are not at all similar to each other.

Consider the case that the myoglobin structure above is stored in the geometric suffix tree as a representative structure, and we want to find all the representative structures whose RSSDs to the rhodopsin structure above is within 20.0Å. Then we must incrementally compare these prefix structures up to 12 atoms. It means that we have to meaninglessly compute RSSDs 12 times, though these two structures are not at all similar to each other.

Let $\pi = \{\pi_1, \pi_2, \ldots, \pi_k\}$ be some permutation of length k. For a structure $P = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n\}$ such that $n \geq k$, consider a new structure $H_\pi(P) = \{\boldsymbol{p}_{\pi_1}, \boldsymbol{p}_{\pi_2}, \ldots, \boldsymbol{p}_{\pi_k}, \boldsymbol{p}_{k+1}, \boldsymbol{p}_{k+2}, \ldots, \boldsymbol{p}_n\}$, which we call the prefix-shuffled structure of P by π.

In Figure 2, the dotted line noted as 'Random' shows the RSSDs between the prefix substructures of the prefix-shuffled structures of the same two structures (a myoglobin and a rhodopsin) with a randomly-generated permutation of length 30.[1] In other words, we compare $H_\pi(P)$ and $H_\pi(Q)$ instead of P and Q. According to the figure, the RSSD exceeds 20.0Å if the prefix substructure length becomes larger than 7, which is much smaller than the '12' in the previous 'Normal' case. It is a very reasonable result, because the distances between two adjacent atoms in the prefix-shuffled structure is often much larger than those in the original structure. Based on these observations, we consider that we may be able to improve the query performance by shuffling the structures with some appropriate permutation (both for the database and the query). The new data structure proposed in the next section is based on this intuition.

[1] The permutation we used here is $\{3, 25, 12, 29, 2, 13, 19, 16, 17, 10, 11, 9, 7, 1, 8, 18, 26, 27, 23, 5, 28, 15, 21, 20, 24, 14, 30, 22, 4, 6\}$.

4 Prefix-Shuffled Geometric Suffix Trees

We define the *prefix-shuffled geometric suffix tree* (*PSGST* for short) for a structure P as the geometric trie over all the prefix-shuffled suffix substructures of P by some permutation π (*i.e.*, $\{H_\pi(P[i..n])|1 \leq i \leq n - |\pi| + 1\}$). The memory requirement for storing the PSGST is $O(n)$ (same as the geometric suffix tree). Recall that the geometric suffix tree is built by just adding each suffix substructures one by one. The PSGSTs can also be built in the same way as the geometric suffix trees, which requires $O(n^2)$ time. Moreover, we can search for substructures that is similar to Q by just searching for representative structures that is similar to the prefix-shuffled query $H_\pi(Q)$ on the PSGST, if the length of Q is not smaller than the length of π. In this paper, we do not deal with queries which are shorter than the permutation π.

To construct the PSGSTs, we need some appropriate permutation of a given length. A random permutation can be used for this purpose. A uniform random permutation of length k can be generated in $O(k)$ time by iteratively swapping each position i ($1 \leq i \leq k$, in increasing order) of a list $\{1, 2, \ldots, k\}$ with a randomly chosen position among positions j such that $j \geq i$ (see [3] for details).

Other than the random permutations, the following permutation can also be used. A permutation $\pi = \{\pi_1, \pi_2, \ldots, \pi_k\}$ is called a *furthest permutation* if it satisfies $\pi_1 = 1$, $\pi_2 = k$, and $\min_{\ell<i} |\pi_i - \pi_\ell| \geq \min_{\ell<i} |\pi_j - \pi_\ell|$ for any i and j such that $i < j$. We call it 'furthest' because π_i is furthest from $\{\pi_1, \ldots, \pi_{i-1}\}$ among $\pi_i, \pi_{i+1}, \ldots, \pi_k$. For example, $\{1, 9, 5, 3, 7, 2, 4, 6, 8\}$ is a furthest permutation. We can assume that the distance between two atoms $\boldsymbol{p}_{\pi_i}$ and $\boldsymbol{p}_{\pi_j}$ would be large if $|\pi_i - \pi_j|$ is large. Therefore we consider the furthest permutation might be suitable for the PSGSTs.

The furthest permutation of length k can also be computed in $O(k)$ time with a bit operation technique as follows. To ease discussion, we first assume that $k - 1$ is a power of 2 and let $d = \log_2(k - 1)$. Let $rev_d(x)$ be a function that reverses the last d bit of x. For example, $rev_3(3) = rev_3(011(2)) = 110(2) = 6$. The function $rev_d(x)$ can be computed in $O(1)$ time.[2] Then consider a permutation $\pi^{(k)}$ of length k where $\pi_1^{(k)} = 1$, $\pi_2^{(k)} = k$, and $\pi_i^{(k)} = rev_d(i - 2) + 1$ for $i \geq 3$. It is the furthest permutation of length k, and it can be computed in $O(k)$ time. In case that $k-1$ is not a power of 2, let k' be the smallest power of 2 that is not smaller than k, and construct the furthest permutation $\pi^{(k'+1)}$ with the above method. Then the furthest permutation of length k can be obtained by just removing numbers larger than k from $\pi^{(k'+1)}$, which requires only $O(k)$ time.

[2] We can compute $rev_d(x)$ by using a pre-computed table of the values of $rev_{\lceil d/c \rceil}(x)$ for $0 \leq x < 2^{\lceil d/c \rceil}$, where c is an appropriate constant positive integer. If we use appropriate c, the table size must be reasonably small, even if k (*i.e.*, $2^d + 1$) is very large. But even without such table, it takes only $O(\log d)$ time to compute $rev_d(x)$ with the basic bit operations of `AND`, `OR`, and `SHIFT`, and consequently the total computing time is still $O(k \log \log k)$ time. If k is a 32-bit (or even a 64-bit) integer, we can assume it as linear time. Note that it is very easy to design a digital circuit that computes $rev_d(x)$ in constant time.

Table 1. Time (in second) for constructing a geometric suffix tree and PSGSTs. The 'Random' columns shows the average/minimum/maximum construction time of 100 PSGSTs constructed with different permutations. The 'Furthest' column shows the construction time for the PSGST constructed with the furthest permutation.

	GST	PSGST			
		Random			Furthest
		Average	Minimum	Maximum	
Time (sec)	39.10	37.26	35.87	38.65	37.89

Thus we conclude that the total computation time for constructing the furthest permutation of length k is $O(k)$.

In the next section, we will show through experiments how well our simple strategy works for 3-D substructure search.

5 Experiments

In this section, we demonstrate the performance of the PSGSTs. All the experiments are done on a Sun Fire 15K super computer with 288 GB memory and 96 UltraSPARC III Cu CPUs running at 1.2GHz.[3] As a data for experiments, we used a set of 228 myoglobin or myoglobin-related PDB data files containing 275 protein structures, which is same as the set used in the experiments by [12]. The total number of amino acids in the protein set is 41,719.

At first, we compared the construction time of PSGSTs against the construction time of the geometric suffix trees, by setting the RSSD bound $b_{RSSD} = 20.0$Å (Table 1). In the table, the 'GST' column shows the construction time of the geometric suffix tree against the myoglobin database. Next, we constructed 100 PSGSTs with different random permutations of length 50.[4] The 'Random' column shows the average, minimum, and maximum construction time among these 100 experiments. They are a little faster than the case of the geometric suffix tree, but it is not much different. We also did experiments by using the furthest permutation of length 50. The 'Furthest' column shows the result. The result is almost the same as the average of the results of random permutations. We assume these results are very reasonable, as there is no difference between the algorithms for the PSGSTs and the geometric suffix trees except for the prefix shuffling.

We next examined the query speed of the above 101 PSGSTs (*i.e.*, the 100 PSGSTs constructed with different random permutations, and the one constructed with the furthest permutation) and the geometric suffix tree (Table 2). We used two protein substructures as queries: (a) A substructure from the 20th amino acid to the 69th amino acid of the backbone structure of a rhodopsin[5]

[3] We used only one CPU for each experiment.

[4] We used the Mersenne-Twister [9] for generating random numbers.

[5] As seen in section 3, rhodopsins have nothing to do with myoglobins, and their structures are totally different.

Table 2. Time (in second) for queries on the geometric suffix trees and the PSGSTs. In (a), we used as a query a protein structure unrelated to any of the structures in a myoglobin structure database. In (b), by contrast, we used a myoglobin structure that is included in the same database.

(a) A rhodopsin query against the myoglobin database.

b_{RMSD} (Å)			0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0
#hits			0	0	0	0	0	0	0	0	0	0
GST			0.0207	0.0825	0.1736	0.2567	0.3306	0.3960	0.4554	0.5146	0.5726	0.6321
PS-GST	Random	avg	0.0028	0.0063	0.0127	0.0241	0.0428	0.0716	0.1130	0.1681	0.2379	0.3244
		min	0.0002	0.0008	0.0018	0.0037	0.0080	0.0185	0.0372	0.0679	0.1053	0.1607
		max	0.0167	0.0461	0.0866	0.1241	0.1621	0.1986	0.2350	0.3020	0.4008	0.5130
	Furthest		0.0013	0.0022	0.0044	0.0081	0.0332	0.0576	0.1012	0.1605	0.2423	0.0163

(b) A myoglobin query against the myoglobin database.

b_{RMSD} (Å)			0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0
#hits			1	1	4	9	19	26	33	44	86	142
GST			0.0654	0.1065	0.1547	0.2333	0.3145	0.3994	0.4662	0.5473	0.7101	0.7859
PS-GST	Random	avg	0.0583	0.0887	0.1219	0.1688	0.2372	0.2960	0.3519	0.4287	0.5443	0.6590
		min	0.0472	0.0712	0.0986	0.1243	0.1889	0.2445	0.2958	0.3476	0.4480	0.5608
		max	0.0778	0.1143	0.1580	0.2180	0.2984	0.3624	0.4359	0.5174	0.6835	0.7871
	Furthest		0.0615	0.0930	0.1637	0.2122	0.2633	0.3164	0.3775	0.4829	0.5796	0.7463

(named 1F88) obtained from the PDB, and (b) A substructure from the 20th amino acid to the 69th amino acid of the backbone structure of a myoglobin (named 103M), which is contained in the myoglobin database we used for constructing the geometric suffix trees and the PSGSTs. Note that these queries are same as those used in [12]. For each query, we searched for similar substructures with 10 different settings of the RMSD bound (b_{RMSD}). In the table, the '#hits' rows show the numbers of similar structures obtained with the designated b_{RMSD} settings, the 'GST' rows show the query time (in second) on the geometric suffix tree, and the 'PSGST' rows show the query time (in second) on the PSGSTs. In the 'PSGST' rows, the 'Random' rows show the average/minimum/maximum query time among the 100 PSGSTs constructed with different random permutations, while the 'Furthest' rows show the query time on the PSGST constructed with the furthest permutation.

In the experiment (a), the PSGST outperforms the geometric suffix tree in all the 101 cases. The PSGSTs constructed with random permutations perform about 1.9–13 times better than the geometric suffix tree in average. Moreover, the PSGSTs perform more than 100 times better than the geometric suffix tree in the best case. If we use the furthest permutation, the PSGST performs about 2.6–37.5 times better than the geometric suffix tree. The results by the furthest permutation is better than the average of results by random permutations, but it is not the best one among the 101 permutations we tried.

Consider a Figure 2-like graph for two similar structures. In this case, the RSSD will not go up until the end of the structure. Thus, we can easily imagine

that the PSGSTs are not so efficient if the database has many structures similar to the query, which can be seen in the experiment (b). But, according to the table, the PSGST outperforms the geometric suffix tree in most cases. If we use a random permutation, the PSGST performs about 1.5 times better than the geometric suffix tree in average. If we use the furthest permutation, the PSGST outperforms the geometric suffix tree in all the cases but 1 case. All in all, we can conclude that the PSGST outperforms the geometric suffix tree.

6 Discussion

We proposed a new data structure based on the geometric suffix tree, which we call the prefix-shuffled geometric suffix tree (PSGST). The PSGSTs show higher query performance than the geometric suffix trees in most cases, though the construction time is almost the same. In the best case, a query on a PSGST is more than 100 times faster than the same query on the geometric suffix tree.

Several tasks remain as future work. The PSGST performs well especially when there are not many substructures similar to the query in the database. It means that the PSGST can be used as a very powerful filtering tool for some other more flexible similarity search algorithms on 3-D structures, which is one of the future tasks. Another future task is finding gapped 3-D motifs of proteins by using the PSGST. We do not know how to get the optimal permutation for the PSGST, which is an open problem. On PSGSTs, we cannot search for queries shorter than the permutation used for constructing the PSGST. It is also an open problem how to smartly deal with such short queries on PSGSTs.

Acknowledgement

All the computational experiments in this research were done on the Super Computer System, Human Genome Center, Institute of Medical Science, University of Tokyo.

References

1. Arun, K.S., Huang, T.S., Blostein, S.D.: Least-squares fitting of two 3-D point sets. IEEE Trans Pattern Anal. Machine Intell. 9, 698–700 (1987)
2. Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N., Bourne, P.E.: The protein data bank. Nucl. Acids Res. 28, 235–242 (2000)
3. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms, 2nd edn. MIT Press, Cambridge (2001)
4. Eggert, D.W., Lorusso, A., Fisher, R.B.: Estimating 3-D rigid body transformations: a comparison of four major algorithms. Machine Vision and Applications 9, 272–290 (1997)
5. Eidhammer, I., Jonassen, I., Taylor, W.R.: Structure Comparison and Structure Patterns. J. Computational Biology 7(5), 685–716 (2000)

6. Farach, M.: Optimal suffix tree construction with large alphabets. In: Proc. 38th IEEE Symp. Foundations of Computer Science, pp. 137–143. IEEE Computer Society Press, Los Alamitos (1997)
7. Golub, G.H., Van Loan, C.F.: Matrix Computation, 3rd edn. John Hopkins University Press (1996)
8. Gusfield, D.: Algorithms on strings, trees, and sequences: computer science and computational biology. Cambridge University Press, Cambridge (1997)
9. Matsumoto, M., Nishimura, T.: A nonempirical test on the weight of pseudorandom number generators. In: Fang, K.T., et al. (eds.) Monte Carrlo and Quasi-Monte Carlo Methods 2000, pp. 381–395. Springer, Heidelberg (2002)
10. McCreight, E.M.: A space-economical suffix tree construction algorithm. J. ACM. 23, 262–272 (1976)
11. Schwartz, J.T., Sharir, M.: Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves. Intl. J. of Robotics Res. 6, 29–44 (1987)
12. Shibuya, T.: Geometric Suffix Tree: A New Index Structure for Protein 3-D Structures. In: Lewenstein, M., Valiente, G. (eds.) CPM 2006. LNCS, vol. 4009, pp. 84–93. Springer, Heidelberg (2006)
13. Ukkonen, E.: On-line construction of suffix-trees. Algorithmica 14, 249–260 (1995)
14. Weiner, P.: Linear pattern matching algorithms. In: Proc. 14th Symposium on Switching and Automata Theory, pp. 1–11 (1973)