Searching Protein 3-D Structures in Linear Time

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. Finding similar structures from 3-D structure databases of proteins is becoming more and more important issue in the post-genomic molecular biology. To compare 3-D structures of two molecules, biologists mostly use the RMSD (root mean square deviation) as the similarity measure. We propose new theoretically and practically fast algorithms for the fundamental problem of finding all the substructures of structures in a structure database of chain molecules (such as proteins), whose RMSDs to the query are within a given constant threshold. We first propose a breakthrough linear-expected-time algorithm for the problem, while the previous best-known time complexity was $O(N \log m)$, where N is the database size and m is the query size. For the expected time analysis, we propose to use the random-walk model (or the ideal chain model) as the model of average protein structures. We furthermore propose a series of preprocessing algorithms that enable faster queries. We checked the performance of our linear-expected-time algorithm through computational experiments over the whole PDB database. According to the experiments, our algorithm is 3.6 to 28 times faster than previously known algorithms for ordinary queries. Moreover, the experimental results support the validity of our theoretical analyses.

1 Introduction

3-D structure database searching of molecules, especially proteins, plays a very important role in molecular biology [2,8,10]. For example, if we have proteins whose structures are known but their functions are unknown, we may be able to predict their functions by searching for similar structures whose functions are known, as structurally similar proteins tend to have similar functions. Moreover, more and more protein structures are solved today with the aid of state-of-the-art technologies such as nuclear magnetic resonance (NMR) techniques, as seen in the rapid growth of the PDB database [3]. Thus, faster searching techniques are seriously needed for the molecular structure databases.

A protein is a chain of amino acids. Thus, its structure can be represented by a sequence of 3-D coordinates, each of which corresponds to the position of a specified atom (the C_α atom is usually used) of each amino acid. Such molecules are called chain molecules. There are also many other important chain molecules in living cells, such as DNAs, RNAs and glycans. The RMSD (root mean square

S. Batzoglou (Ed.): RECOMB 2009, LNCS 5541, pp. 1–15, 2009.

deviation) [1,7,12,13,16,19] is the fundamental measure to determine the geometric similarity between two same-length sequences of 3-D coordinates. It has been widely used not only for molecular structure comparison, but also for various problems in various fields, such as computer vision and robotics. It is defined as the square root of the minimum value of the average squared distance between each pair of corresponding atoms, over all the possible rotations and translations (see section 2.2). In this paper, we consider one of the most fundamental RMSD-related problems as follows.

Problem. Given a structure database $\mathcal{D}$ of chain molecules and a query structure $\mathbf{Q}$, find all the substructures of the structures in $\mathcal{D}$ whose RMSDs to $\mathbf{Q}$ are at most a given fixed threshold c, without considering any insertions or deletions.

In general, c should be set to a fixed constant proportional to the distance between two adjacent atoms of the chain molecules. In the case of proteins, the distance between two adjacent C_α atoms is around 3.8Å, while two protein structures are said to be similar to each other if their RMSD is smaller than 1 or 2Å.

Our results. The best-known worst-case/expected time complexity of the problem was $O(N \log m)$ [16,19], where N is the database size (*i.e.*, the sum of the lengths of all the structures in the database) and m is the query size. We propose the first linear-expected-time (*i.e.*, $O(N)$) algorithm. To analyze the expected time of the algorithm, we give an assumption that the structures in the database follow a model called the random-walk model (see section 2.4). We also propose several preprocessing algorithms that enable faster queries. We first propose an $O(N \log N)$-time and $O(N)$-space preprocessing algorithm that enables $O(m+N/\sqrt{m})$-expected-time query, for queries of a fixed length. We next extend it to an $O(N \log^2 N)$-time and $O(N \log N)$-space preprocessing algorithm that enables the same $O(m + N/\sqrt{m})$-expected-time query, for queries of arbitrary lengths. We also propose an $O(N \log N)$-time and $O(N)$-space preprocessing algorithm that enables $O(\frac{N}{\sqrt{m}} + m \log(N/m))$-expected-time query, for queries of arbitrary lengths.

We also examine the performance of our algorithms by computational experiments on the whole PDB database. Our linear-time algorithm is much faster than any of previous algorithms, *i.e.*, 3.6 to 28 times faster to search for substructures whose RMSDs are at most 1Å. Moreover, no inconsistency is observed between the theoretical results and the experimental results, which means our random-walk assumption is very reasonable for analyses of algorithms for protein structure database search.

The organization of this paper. In section 2, we describe the basic definitions and related previous work as preliminaries. In section 3, we propose an $O(N\sqrt{m})$ algorithm for the problem above, where N is the database size and m is the length of the query. We next improve it to obtain the linear-time algorithm in section 4. In section 5, we further extend our algorithms for faster queries after preprocessing. In section 6, we examine the performance of our algorithms against the PDB database. In section 7, we conclude our results.

2 Preliminaries

2.1 Notations and Definitions

A chain molecule is represented like $\mathbf{S} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n\}$, where $\boldsymbol{s}_i$ denotes the 3-D coordinates of the i-th atom. The length n of $\mathbf{S}$ is denoted by $|\mathbf{S}|$. A structure $\mathbf{S}[i..j] = \{\boldsymbol{s}_i, \boldsymbol{s}_{i+1}, \ldots, \boldsymbol{s}_j\}$ $(1 \leq i \leq j \leq n)$ is called a substructure of $\mathbf{S}$. $R \cdot \mathbf{S}$ denotes the structure $\mathbf{S}$ rotated by the rotation matrix R, *i.e.*, $R \cdot \mathbf{S} = \{R\boldsymbol{s}_1, R\boldsymbol{s}_2, \ldots, R\boldsymbol{s}_n\}$. $\boldsymbol{v}^t$ denotes the transpose of the vector $\boldsymbol{v}$ and A^T denotes the transpose of the matrix A. $trace(A)$ denotes the trace of the matrix A. $|\boldsymbol{v}|$ denotes the norm of the vector $\boldsymbol{v}$. $\mathbf{0}$ denotes the zero vector. $\langle x \rangle$ denotes the expected value of x. $var(x)$ denotes the variance of x. $Prob(X)$ denotes the probability of X.

In the rest of this paper, we consider that the target database $\mathcal{D}$ consists of one long structure $\mathbf{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_N\}$, and we let $\mathbf{Q} = \{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_m\}$ denote the query structure, where m is supposed to be smaller than N. Our problem is to find all the positions i such that the RMSD (see section 2.2 for its definition) between $\mathbf{P}[i..i+m-1]$ and $\mathbf{Q}$ is at most a given fixed threshold c. An ordinary database may contain more than one structure, but the problem against such databases can be reduced to the problem against databases with only one structure, by concatenating all the database structures into one structure and ignoring substructures that cross over the boundaries of concatenated structures.

2.2 RMSD: The Root Mean Square Deviation

The RMSD (root mean square deviation) [1,7,12,13,16,19] between two 3-D coordinate sequences $\mathbf{S} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n\}$ and $\mathbf{T} = \{\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_n\}$ is defined as the minimum value of $E_{R,\boldsymbol{v}}(\mathbf{S}, \mathbf{T}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} |\boldsymbol{s}_i - (R \cdot \boldsymbol{t}_i + \boldsymbol{v})|^2}$ over all the possible rotation matrices R and translation vectors $\boldsymbol{v}$. Let $RMSD(\mathbf{S}, \mathbf{T})$ denote the minimum value, and let $\hat{R}(\mathbf{S}, \mathbf{T})$ and $\hat{\boldsymbol{v}}(\mathbf{S}, \mathbf{T})$ denote the rotation matrix and the translation vector that minimizes $E_{R,\boldsymbol{v}}(\mathbf{S}, \mathbf{T})$. $RMSD(\mathbf{S}, \mathbf{T})$, $\hat{R}(\mathbf{S}, \mathbf{T})$ and $\hat{\boldsymbol{v}}(\mathbf{S}, \mathbf{T})$ can be computed in $O(n)$ time as follows.

If the rotation matrix R is fixed, $E_{R,\boldsymbol{v}}(\mathbf{S}, \mathbf{T})$ is known to be minimized when the centroid (center of mass) of $R \cdot \mathbf{T}$ is translated to the centroid of $\mathbf{S}$ by the translation vector $\boldsymbol{v}$, regardless of what the rotation matrix R is. It means that $\hat{\boldsymbol{v}}(\mathbf{S}, \mathbf{T})$ can be computed in linear time if we are given $\hat{R}(\mathbf{S}, \mathbf{T})$. Moreover, it also means that the problem of computing the RMSD can be reduced to a problem of finding R (*i.e.*, $\hat{R}(\mathbf{S}, \mathbf{T})$) that minimizes $E'_R(\mathbf{S}, \mathbf{T}) = \sum_{i=1}^{n} |\boldsymbol{s}_i - R \cdot \boldsymbol{t}_i|^2$, by translating both $\mathbf{S}$ and $\mathbf{T}$ so that both of their centroids are moved to the origin of the coordinates, which can be done in linear time.

After translating both structures so that both of their centroids are moved to the origin, we can compute $\hat{R}(\mathbf{S}, \mathbf{T})$ in linear time as follows [1,12,13].[1] Let $J = \sum_{i=1}^{n} \boldsymbol{s}_i \cdot \boldsymbol{t}_i{}^t$. Clearly, J can be computed in $O(n)$ time. Then $E'_R(\mathbf{S}, \mathbf{T})$ can

[1] Here, $\mathbf{S}$ and $\mathbf{T}$ are the translated structures which are different from the original $\mathbf{S}$ and $\mathbf{T}$ in the original problem (that optimizes both rotation and translation).

be described as $\sum_{i=1}^{n} ({\boldsymbol{s}_i}^t \boldsymbol{s}_i + {\boldsymbol{t}_i}^t \boldsymbol{t}_i) - 2 \cdot trace(R \cdot J)$, and $trace(R \cdot J)$ is maximized when $R = VU^T$, where $U \Lambda V$ is the singular value decomposition (SVD) of J. Thus $\hat{R}(\mathbf{S}, \mathbf{T})$ can be obtained from J in constant time, as J is a 3×3 matrix and the SVD can be computed in $O(d^3)$ time for a $d \times d$ matrix [11]. Note that there are degenerate cases where $det(VU^T) = -1$, which means that VU^T is a reflection matrix. See [1,7,12] for the details of the degenerate cases. Finally, we can compute the RMSD in linear time once we have obtained $\hat{R}(\mathbf{S}, \mathbf{T})$. In total, we can compute the RMSD in $O(n)$ time.

2.3 Previous Best-Known Searching Algorithms

According to the previous section, we can compute the RMSD between any substructure $\mathbf{P}[i..i+m-1]$ and the query $\mathbf{Q}$ in $O(m)$ time. Consequently, we can solve our problem in $O(Nm)$ time by checking the RMSDs between the query and all the $O(N)$ substructures of length m in the database.

Schwartz and Sharir [16] proposed a more sophisticated approach for the problem that solves it in $O(N \log N)$ time, based on the convolution technique using the FFT (fast Fourier transform) [5]. Shibuya [19] also proposed a different algorithm with the same time complexity, also based on the convolution technique. These algorithms are not faster than the naive algorithm when $N \gg m$. But this time bound can be easily improved to $O(N \log m)$ as follows. Break $\mathbf{P}$ into $O(N/m)$ substructures of length $m + O(m)$ each of which overlaps with its adjacent fragment with overlap length $m - 1$. Then our problem can be solved in $O(N \log m)$ time by applying the above $O(N \log N)$-time algorithm against each fragment. The expected time complexity of these algorithms are all the same as their worst-case time complexity, and no algorithm with better expected time complexity is known. But note that the above FFT-based $O(N \log m)$-time algorithm is not practically faster than the naive $O(Nm)$-time algorithm in case m is not large enough, and it is rarely used in practice.

For the problem, a linear-size indexing data structure called the geometric suffix tree [17,18] is known to enable practically faster query than the above algorithms. But its worst-case query time complexity is still $O(Nm)$, while we need $O(N^2)$ time to construct the data structure. In fact, there have been no known indexing algorithms whose theoretical query time complexity is smaller than the above $O(N \log m)$ bound.

2.4 The Random-Walk Model for Chain Molecule Structures

The *random-walk model* for chain molecule structures is a simple but useful model for analyzing their behavior [4,6,9,15]. The model is also called the *freely-jointed chain model* or the *ideal chain model*. In the model, we assume that the structure of a chain molecule is constructed as a result of a random walk in 3-D space. It is useful in various analyses in molecular physics, as it reflects properties of structures of real chain molecules very well [4].

Consider a chain molecule $\mathbf{S} = \{\boldsymbol{s}_0, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n\}$ of length $n + 1$, in which the distance between two adjacent atoms is fixed to some constant ℓ. Note that

the length between two adjacent C_α atoms in a protein structure is constantly 3.8Å, as mentioned in section 1. In the random-walk model, a bond between two adjacent atoms, *i.e.*, $\boldsymbol{b}_i = \boldsymbol{s}_{i+1} - \boldsymbol{s}_i$, is considered as a random vector that satisfies $|\boldsymbol{b}_i| = \ell$, and $\boldsymbol{b}_i$ is independent from $\boldsymbol{b}_j$ for any i and j $(i \neq j)$. If n is large enough, the distribution of the end-to-end vector $\boldsymbol{s}_n - \boldsymbol{s}_0$ is known to converge to the Gaussian distribution in 3-D space, in which $\langle \boldsymbol{s}_n - \boldsymbol{s}_0 \rangle = \boldsymbol{0}$ and $\langle |\boldsymbol{s}_n - \boldsymbol{s}_0|^2 \rangle = n \cdot \ell^2$. In the distribution, the probability (or probability density) that $\boldsymbol{s}_n - \boldsymbol{s}_0$ is located at some position (x, y, z) is $W_{n,\ell}(x, y, z)dxdydz = (\frac{3}{2\pi n\ell^2})^{\frac{3}{2}} e^{-3(x^2+y^2+z^2)/2n\ell^2} dxdydz$.

In our theoretical analysis in later sections, we will give an assumption that the structures in the target database follow the random-walk model. Our experimental results on the PDB database in section 6 show high consistency with our theoretical analyses based on the model.

3 An $O(N\sqrt{m})$ Algorithm

In this section, we propose an algorithm that will be the basis for our algorithms in later sections. The algorithm given in section 3.2 achieves $O(N\sqrt{m})$ time under the random-walk assumption, by filtering out most of the dissimilar substructures (*i.e.*, substructures with RMSDs larger than the given threshold c) before computing the actual RMSDs, based on a measure described in section 3.1. Its time complexity will be analyzed in section 3.3.

3.1 An Efficiently-Computable Lower Bound for the RMSD

We first propose a nontrivial, but easily-computable lower bound for the RMSD between any two structures with the same length. Let $\mathbf{U}^{left}$ denote $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{\lfloor k/2 \rfloor}\}$ and $\mathbf{U}^{right}$ denote $\{\boldsymbol{u}_{\lfloor k/2 \rfloor+1}, \boldsymbol{u}_{\lfloor k/2 \rfloor+2}, \ldots, \boldsymbol{u}_{2\cdot\lfloor k/2 \rfloor}\}$ for a structure $\mathbf{U} = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_k\}$. Let $G(\mathbf{U})$ denote the centroid (center of mass) of the structure $\mathbf{U}$, *i.e.*, $G(\mathbf{U}) = \frac{1}{k}\sum_{i=1}^{k} \boldsymbol{u}_i$. Let $F(\mathbf{U})$ denote $|G(\mathbf{U}^{left}) - G(\mathbf{U}^{right})|/2$, which means the half of the distance between the centroids of $\mathbf{U}^{left}$ and $\mathbf{U}^{right}$. For any two structures $\mathbf{S}$ and $\mathbf{T}$ with the same length n, we define $D(\mathbf{S}, \mathbf{T})$ as $|F(\mathbf{S}) - F(\mathbf{T})|$ if n is an even integer. If n is an odd integer, we define $D(\mathbf{S}, \mathbf{T})$ as $\sqrt{\frac{n-1}{n}}|F(\mathbf{S}) - F(\mathbf{T})|$. From now on, we prove that $D(\mathbf{S}, \mathbf{T})$ is a lower bound of the RMSD between $\mathbf{S}$ and $\mathbf{T}$.

Let $\boldsymbol{s}'_i = \boldsymbol{s}_i - G(\mathbf{S})$, and $\boldsymbol{t}'_i = \boldsymbol{t}_i - G(\mathbf{T})$. In case n is an even integer, we prove that $D(\mathbf{S}, \mathbf{T})$ is always smaller than or equal to $RMSD(\mathbf{S}, \mathbf{T})$, as follows:

$$RMSD(\mathbf{S}, \mathbf{T}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} |\boldsymbol{s}'_i - \hat{R}(\mathbf{S}, \mathbf{T}) \cdot \boldsymbol{t}'_i|^2} \geq \frac{1}{n}\sum_{i=1}^{n} |\boldsymbol{s}'_i - \hat{R}(\mathbf{S}, \mathbf{T}) \cdot \boldsymbol{t}'_i|$$

$$\geq \frac{1}{n}\Big|\sum_{i=1}^{n/2}\{\boldsymbol{s}'_i - \hat{R}(\mathbf{S}, \mathbf{T}) \cdot \boldsymbol{t}'_i\}\Big| + \frac{1}{n}\Big|\sum_{i=n/2+1}^{n}\{\boldsymbol{s}'_i - \hat{R}(\mathbf{S}, \mathbf{T}) \cdot \boldsymbol{t}'_i\}\Big|$$

$$
\begin{aligned}
&= |G(\mathbf{S}^{left}) - G(\mathbf{S}) - \hat{R}(\mathbf{S},\mathbf{T}) \cdot \{G(\mathbf{T}^{left}) - G(\mathbf{T})\}|/2 \\
&\quad + |G(\mathbf{S}^{right}) - G(\mathbf{S}) - \hat{R}(\mathbf{S},\mathbf{T}) \cdot \{G(\mathbf{T}^{right}) - G(\mathbf{T})\}|/2 \\
&= |G(\mathbf{S}^{left}) - G(\mathbf{S}^{right}) - \hat{R}(\mathbf{S},\mathbf{T}) \cdot \{G(\mathbf{T}^{left}) - G(\mathbf{T}^{right})\}|/2 \\
&\geq |\{|G(\mathbf{S}^{left}) - G(\mathbf{S}^{right})| - |G(\mathbf{T}^{left}) - G(\mathbf{T}^{right})|\}|/2 \\
&= D(\mathbf{S},\mathbf{T}). \qquad (1)
\end{aligned}
$$

Note that we used the fact that $G(\mathbf{S}) = \{G(\mathbf{S}^{left}) + G(\mathbf{S}^{right})\}/2$ and $G(\mathbf{T}) = \{G(\mathbf{T}^{left}) + G(\mathbf{T}^{right})\}/2$ in the above.

In case n is an odd integer, we prove the same as follows:

$$
\begin{aligned}
RMSD(\mathbf{S},\mathbf{T}) &\geq \sqrt{\frac{n-1}{n}} RMSD(\mathbf{S}^-,\mathbf{T}^-) \\
&\geq \sqrt{\frac{n-1}{n}} D(\mathbf{S}^-,\mathbf{T}^-) = D(\mathbf{S},\mathbf{T}), \qquad (2)
\end{aligned}
$$

where $\mathbf{S}^-$ denotes $\{s_1, s_2, \ldots, s_{n-1}\}$, and $\mathbf{T}^-$ denotes $\{t_1, t_2, \ldots, t_{n-1}\}$.

If the above lower bound can be computed very efficiently, we may solve our problem (presented in section 1) more efficiently by filtering out hopelessly dissimilar substructures before computing the actual RMSD value. In fact, we can compute the above lower bound $D(\mathbf{P}[i..i+m-1],\mathbf{Q})$ for all the positions i in linear time, as follows. For any m, we can compute $G(\mathbf{P}[i..i+\lfloor m/2 \rfloor - 1])$ for all the positions i such that $1 \leq i \leq N - \lfloor m/2 \rfloor + 1$ in $O(N)$ time, as $G(\mathbf{P}[i..i+\lfloor m/2 \rfloor - 1] = G(\mathbf{P}[i-1..i+\lfloor m/2 \rfloor - 2]) - \frac{1}{\lfloor m/2 \rfloor}(\boldsymbol{p}_{i-1} - \boldsymbol{p}_{i+\lfloor m/2 \rfloor - 1})$. It means that $F(\mathbf{P}[i..i+m-1],\mathbf{Q})$ and consequently $D(\mathbf{P}[i..i+m-1],\mathbf{Q})$ can be computed for all the positions i such that $1 \leq i \leq N-m+1$, also in $O(N)$ time.

3.2 The Algorithm

Our basic algorithm is simple. It uses the above lower bound to filter out some (hopefully most) of the substructures in the database before the time-consuming RMSD computation, as follows.

Algorithm 1

```
1  Compute D_i = D(P[i..i+m-1], Q)
       for all i such that 1 ≤ i ≤ N-m+1.
2  for (all i such that 1 ≤ i ≤ N-m+1) {
3    if (D_i ≤ c) {
4      if (RMSD(P[i..i+m-1], Q) ≤ c)
5        { output("P[i..i+m-1] is similar to the query Q.") }
6    }
7  }
```

The above algorithm is valid, *i.e.*, it enumerates all the positions of the substructures whose RMSDs to the query are at most c, because $D_i = D(\mathbf{P}[i..i+m-1], \mathbf{Q})$ is always smaller than or equal to $RMSD(\mathbf{P}[i..i+m-1], \mathbf{Q})$. Let the number of times of the RMSD computation in line 4 be $N'(\leq N)$. Then, the time complexity of the above algorithm is $O(N+N'm)$, as the line 1 of the algorithm requires only $O(N)$ time according to the discussion in section 3.1. In the next section, we will prove that $\langle N' \rangle$ is in $O(N/\sqrt{m})$, if the structures in the database follow the random-walk model.

3.3 Computational Time Analysis

Consider a structure $\mathbf{S} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_{2n}\}$ of length $2n$ that follows the random-walk model. In this section, we let the distance between two adjacent atoms (ℓ in section 2.4) be 1 without loss of generality, *i.e.*, we consider the distance between two adjacent atoms as the unit of distance. Then $\boldsymbol{s}_i$ can be represented as $\boldsymbol{s}_1 + \sum_{j=1}^{i-1} \boldsymbol{b}_j$, where $\boldsymbol{b}_i$ is an independent random vector that satisfies $|\boldsymbol{b}_i| = 1$. Let $H(\mathbf{S}) = G(\mathbf{S}^{left}) - G(\mathbf{S}^{right})$. Notice that $F(\mathbf{S}) = |H(\mathbf{S})/2|$. Then the following equation holds:

$$H(\mathbf{S}) = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{s}_1 + \sum_{j=1}^{i-1}\boldsymbol{b}_j) - \frac{1}{n}\sum_{i=n+1}^{2n}(\boldsymbol{s}_1 + \sum_{j=1}^{i-1}\boldsymbol{b}_j)$$

$$= -\{\sum_{i=1}^{n}\frac{i}{n}\cdot\boldsymbol{b}_i + \sum_{i=n+1}^{2n-1}\frac{2n-i}{n}\cdot\boldsymbol{b}_i\}.[1mm] \quad (3)$$

Let $\boldsymbol{b}'_i$ denote $\frac{i}{n}\cdot\boldsymbol{b}_i$ if $i \leq n$ and $\frac{2n-i}{n}\cdot\boldsymbol{b}_i$ if $i > n$. Then $H(\mathbf{S})$ can be described as $\sum_{i=1}^{2n}\boldsymbol{b}'_i$. Let z_i denote the z coordinate of $\boldsymbol{b}_i$ and z'_i denote the z coordinate of $\boldsymbol{b}'_i$. It is easy to see that $\langle z_i \rangle = 0$ and $var(z_i) = 1/3$, as $\boldsymbol{b}_i$ is a random vector that satisfies $|\boldsymbol{b}_i| = 1$. Let $M_n = \sum_{i=1}^{2n}\langle|z'_i - \langle z'_i\rangle|\rangle^{2+\delta} / \sqrt{\sum_{i=1}^{2n} var(z'_i)}^{2+\delta}$, where δ is some positive constant. According to Lyapunov's central limit theorem [14], the distribution of $\sum_i^{2n} z'_i$ converges to the Gaussian distribution, if M_n converges to 0 as n grows up to infinity for some δ such that $\delta > 0$. It can be proved as follows:

$$M_n = \sum_{i=1}^{2n}\langle|z'_i|\rangle^{2+\delta} / \sqrt{\sum_{i=1}^{2n}\langle|z'_i|\rangle^2}^{2+\delta} \leq \sum_{i=1}^{2n}\langle|z'_i|\rangle^2 / \sqrt{\sum_{i=1}^{2n}\langle|z'_i|\rangle^2}^{2+\delta}$$

$$= \{\sum_{i=1}^{2n}\langle|z'_i|\rangle^2\}^{-\delta/2} = \{\frac{1}{3n^2}\cdot\{\sum_{i=1}^{n} i^2 + \sum_{i=n+1}^{2n-1}(2n-i)^2\}\}^{-\delta/2}$$

$$= \{\frac{2}{9}n + \frac{1}{9n}\}^{-\delta/2} \ \rightarrow\ 0 \quad (n \rightarrow \infty).[2mm] \quad (4)$$

Hence, we conclude that $\sum_i^{2n} z'_i$ converges to the Gaussian distribution. It also means that $H(\mathbf{S})$ converges to the Gaussian distribution in 3-D space if n

grows up to infinity, as the same discussion can be done for the other two axes (x and y). The variance of $H(\mathbf{S})$ is:

$$\begin{aligned} var(H(\mathbf{S})) &= \langle |H(\mathbf{S})|^2 \rangle - \langle |H(\mathbf{S})| \rangle^2 = \langle |H(\mathbf{S})|^2 \rangle \\ &= \langle | \sum_{i=1}^{n} \frac{i}{n} \cdot \boldsymbol{b}_i + \sum_{i=n+1}^{2n-1} \frac{2n-i}{n} \cdot \boldsymbol{b}_i |^2 \rangle \\ &= \frac{1}{n^2} \{ \sum_{i=1}^{n} i^2 + \sum_{i=n+1}^{2n-1} (2n-i)^2 \} = \frac{2}{3}n + \frac{1}{3n} \approx \frac{2}{3}n, [1mm] \end{aligned} \quad (5)$$

as $\langle \boldsymbol{b}_i \cdot \boldsymbol{b}_j \rangle = 0$ if $i \neq j$. Moreover, it is easy to see that $\langle H(\mathbf{S}) \rangle = \mathbf{0}$. Thus the distribution of $H(\mathbf{S})$ is the same as the distribution of random walks of length $2n/3$. Hence the probability distribution of $H(\mathbf{S})$ is $Z_n(x,y,z)dxdydz = (\frac{9}{4\pi n})^{\frac{3}{2}} e^{-9(x^2+y^2+z^2)/4n} dxdydz$. Consequently, the probability (or probability density) that $|H(\mathbf{S})| = r$ is $Z_n(r)dr = 4\pi r^2 (\frac{9}{4\pi n})^{\frac{3}{2}} e^{-9r^2/4n} dr$. Integrating $Z_n(r)dr$, we obtain $Prob(x \leq |H(\mathbf{S})| \leq y) = \int_{r=x}^{y} Z_n(r)dr$. $Z_n(r)$ takes the maximum value at $r_{max} = \frac{2}{3}\sqrt{n}$ and $Z_n(r_{max}) = 6e^{-1}/\sqrt{\pi n}$. Thus $Prob(x \leq |H(\mathbf{S})| \leq y)$ is at most $(y-x) \cdot Z_n(r_{max}) = 6e^{-1}(y-x)/\sqrt{\pi n}$ for any x and y $(x < y)$.

Therefore, for any structure $\mathbf{T}$ such that $|\mathbf{T}| = |\mathbf{S}| = 2n$, the probability $Prob(|D(\mathbf{S},\mathbf{T})| \leq c) = Prob(F(\mathbf{T}) - c \leq F(\mathbf{S}) \leq F(\mathbf{T}) + c) = Prob(2 \cdot F(\mathbf{T}) - 2c \leq |H(\mathbf{S})| \leq 2 \cdot F(\mathbf{T}) + 2c))$ is at most $4 \cdot c \cdot Z_n(r_{max}) = 24c \cdot e^{-1}/\sqrt{\pi n}$, which is in $O(1/\sqrt{n})$ as c is a fixed constant. Notice that there is no assumption on the structure T in this analysis.

Consequently, as $\sqrt{\frac{m-1}{m}} \approx 1$, the probability $Prob(D_i \leq c)$ in the line 3 of the algorithm in section 3.2 is in $O(1/\sqrt{m})$ if $\mathbf{P}$ follows the random-walk model, no matter what the query structure $\mathbf{Q}$ is. It means that $\langle N' \rangle$ is in $O(N/\sqrt{m})$. Therefore, we conclude that the expected time complexity of the algorithm is $O(N + \langle N' \rangle \cdot m) = O(N\sqrt{m})$, under the assumption that the structures in the database follow the random-walk model.[2] Note that the worst-case time complexity of the algorithm is still $O(Nm)$ as N' can be in $O(N)$ at worst, but it should be rare under the random-walk assumption.

4 The Linear-Time Algorithm

We next improve the algorithm 1 to obtain better expected time complexity. From the definition of the RMSD, we can deduce that

$$\begin{aligned} RMSD(\mathbf{S},\mathbf{T}) &\geq [2mm]\sqrt{t} \cdot \{ (RMSD(\mathbf{S}^{left}, \mathbf{T}^{left}))^2 + \\ &\qquad (RMSD(\mathbf{S}^{right}, \mathbf{T}^{right}))^2 \}^{1/2} \\ &\geq \sqrt{t} \cdot \{ (D(\mathbf{S}^{left}, \mathbf{T}^{left}))^2 + (D(\mathbf{S}^{right}, \mathbf{T}^{right}))^2 \}^{1/2}, \end{aligned} \quad (6)$$

[2] The same discussion can be done in case the query structures, instead of the database structures, follow the random-walk model.

where $t = |\mathbf{S}^{left}|/|\mathbf{S}| = |\mathbf{S}^{right}|/|\mathbf{S}| \approx 1/2$. Let $D^{left}(\mathbf{S}, \mathbf{T}) = \sqrt{t} \cdot D(\mathbf{S}^{left}, \mathbf{T}^{left})$ and $D^{right}(\mathbf{S}, \mathbf{T}) = \sqrt{t} \cdot D(\mathbf{S}^{right}, \mathbf{T}^{right})$. The expression (6) can also be used as a lower bound of the RMSD for another valid filtering algorithm, as follows:

Algorithm 2

```
1   For all i such that 1 ≤ i ≤ N − m + 1, compute
      D'_i = {(D^left(P[i..i + m − 1], Q))^2 +
                (D^right(P[i..i + m − 1], Q))^2}^1/2.
2   for (all i such that 1 ≤ i ≤ N − m + 1) {
3     if (D'_i ≤ c) {
4       if (RMSD(P[i..i + m − 1], Q) ≤ c)
5         { output("P[i..i + m − 1] is similar to the query Q.") }
6     }
7   }
```

The only difference from the Algorithm 1 is the lower bound D'_i used in the line 1. Note that the time complexity of the line 1 is still $O(N)$.

If $D'_i \leq c$ in line 3, both $D^{left}(\mathbf{P}[i..i+m-1], \mathbf{Q})$ and $D^{right}(\mathbf{P}[i..i+m-1], \mathbf{Q})$ must also be at most c. According to the discussion in section 3.3, the two probabilities $Prob(D^{left}(\mathbf{P}[i..i+m-1], \mathbf{Q}) \leq c)$ and $Prob(D^{right}(\mathbf{P}[i..i+m-1], \mathbf{Q}) \leq c)$ are independent and both in $O(1/\sqrt{m})$, under the assumption that $\mathbf{P}$ follows the random-walk model. Thus, $Prob(D'_i \leq c)$ in line 3 must be in $O((1/\sqrt{m})^2) = O(1/m)$. Therefore the expected number of RMSD computations in line 4 should be in only $O(N/m)$, and consequently the expected time complexity spent in the lines 4–6 of the above algorithm is $O(N)$. Thus, the total expected time complexity of the algorithm 2 is $O(N)$ under the random-walk assumption. Note that the worst-case time complexity is still $O(Nm)$, but it should be very rare under the random-walk assumption.

5 Faster Queries after Preprocessing

5.1 Preprocessing for Queries of a Fixed Length

From now on, we further improve the query time complexity by allowing preprocessing on the database structure $\mathbf{P}$. First, we consider the case where each query has the same length m. According to section 3.1, we can compute $F(\mathbf{P}[i..i + w - 1])$ for all i in $O(N)$ time for a fixed value of w. Let L_w be the sorted list of i according to the value of $F(\mathbf{P}[i..i + w - 1])$, which can be obtained in $O(N \log N)$ time. By doing a binary search on L_w, we can find all the i such that $x \leq F(\mathbf{P}[i..i + w - 1]) \leq y$ in $O(\log N + occ)$ time for any x and y, where *occ* is the number of the outputs. Hence, we can list all the i such that $D(\mathbf{S}, \mathbf{P}[i..i + w - 1]) \leq c$ in $O(\log N + occ)$ time for any structure $\mathbf{S}$ of length w by utilizing L_w, where c is some constant and *occ* is the number of outputs, as $F(\mathbf{S}) - c \leq F(\mathbf{P}[i..i + w - 1]) \leq F(\mathbf{S}) + c$ if $D(\mathbf{S}, \mathbf{P}[i..i + w - 1]) \leq c$. Our preprocessing algorithm in this section computes $F(\mathbf{P}[i..i+m'-1])$ for all i and

sorts them to obtain $L_{m'}$, where $m' = \lfloor m/3 \rfloor$, which can be done in $O(N \log N)$ time in total.

Consider yet another lower bound for the RMSD, as follows:[3]

$$\begin{aligned} D''_i &= \sqrt{\frac{m'}{m}} \cdot \{\sum_{j=1}^{3} (D(\mathbf{P}[i + (j-1) \cdot m' .. i + j \cdot m' - 1], \\ &\qquad\qquad \mathbf{Q}[1 + (j-1) \cdot m' .. j \cdot m'])^2\}^{1/2} \\ &\leq \sqrt{\frac{m'}{m}} \cdot \{\sum_{j=1}^{3} (RMSD(\mathbf{P}[i + (j-1) \cdot m' .. i + j \cdot m' - 1], \\ &\qquad\qquad \mathbf{Q}[1 + (j-1) \cdot m' .. j \cdot m'])^2\}^{1/2} \\ &\leq RMSD(\mathbf{P}[i..i + m - 1], \mathbf{Q}). \end{aligned} \tag{7}$$

Notice that $D(\mathbf{P}[i + (j-1) \cdot m' .. i + j \cdot m' - 1], \mathbf{Q}[1 + (j-1) \cdot m' .. j \cdot m']) \leq c\sqrt{\frac{m}{m'}}$ for any j, if $D''_i \leq c$. Let X_j be the set of all positions i such that $D(\mathbf{P}[i + (j-1) \cdot m' .. i + j \cdot m' - 1], \mathbf{Q}[1 + (j-1) \cdot m' .. j \cdot m']) \leq c\sqrt{\frac{m}{m'}}$ (for $1 \leq j \leq 3$). By using $L_{m'}$, we can find all $i \in X_j$ in $O(\log N + |X_j|)$ time for any of $j = 1, 2, 3$ after we have computed $F(\mathbf{Q}[1 + (j-1) \cdot m' .. j \cdot m'])$. Note that $F(\mathbf{Q}[1 + (j-1) \cdot m' .. j \cdot m'])$ for all of $j = 1, 2, 3$ can be computed in $O(m)$ time. Let X be the set of common integers of the three sets X_1, X_2, and X_3. X can be obtained in expected $O(|X_1| + |X_2| + |X_3|)$ time by using a hashing technique. Notice that the positions i such that $RMSD(\mathbf{P}[i..i + m - 1], \mathbf{Q}) \leq c$ must be included in X. For substructures at the positions $i \in X$, we finally have to compute the RMSD to check whether the actual RMSD is at most c, if $D''_i \leq c$. It can be done in at most $O(m \cdot |X|)$ time.

$\langle |X_j| \rangle$ is in $O(N/\sqrt{m})$ under the assumption that $\mathbf{P}$ follows the random-walk model. Moreover, as the structures $\mathbf{P}[i+(j-1)\cdot m' .. i+j\cdot m'-1]$ with different j are independent random walks, $\langle |X| \rangle$ is estimated as $O(N/(\sqrt{m})^3) = O(N/m^{1.5})$. Thus the total expected query time complexity utilizing $L_{m'}$ is $O(m + N/\sqrt{m} + m \cdot N/m^{1.5} + \log N) = O(m + N/\sqrt{m})$.

5.2 Preprocessing for Queries of Arbitrary Lengths

We next consider queries of arbitrary lengths. For such queries, consider computing L_w for all w such that w is a power of 2, *i.e.*, representable as 2^d for some integer d. They can be obtained in $O(N \log^2 N)$ time, as the number of different w is in $O(\log N)$. Let m' be the largest power of 2 such that $3m' \leq m$. Then the inequality (7) also holds for this case. The only difference is that m' is some power of 2 that satisfies $m/6 < m' \leq m/3$, while $m' = \lfloor m/3 \rfloor$ in the previous section. Thus, according to the same discussion as in the previous section, we obtain the same query time complexity, *i.e.*, $O(m + N/\sqrt{m})$. A problem is that

[3] We can consider another linear-expected time algorithm based on the Algorithms 1 and 2 by replacing D_i or D'_i with D''_i. We will examine the performance of the algorithm (denoted as A3 in section 6) through computational experiments in section 6.

the algorithm requires $O(N \log N)$ space to store all the L_w, which might be undesired for huge databases.

5.3 Preprocessing with Linear Space for Queries of Arbitrary Lengths

In this section, we propose another preprocessing algorithm that uses only $O(N)$ space for queries of arbitrary lengths. Consider dividing $\mathbf{P}$ into substructures of length 2^d for each d such that $1 \leq d \leq \log_2 N$. By doing so, we get substructures $\mathbf{P}^{k,d} = \mathbf{P}[(k-1) \cdot 2^d + 1..k \cdot 2^d]$ $(1 \leq k \leq N/2^d)$ for each d. There are only $O(N)$ number of substructures denoted as $\mathbf{P}^{k,d}$, even if we enumerate all the possible k and d. $G(\mathbf{P}^{k,1})$ (see section 3 for its definition) can be computed in constant time for each k. Moreover, $G(\mathbf{P}^{k,d}) = \{G(\mathbf{P}^{2k-1,d-1}) + G(\mathbf{P}^{2k,d-1})\}/2$. Thus, we can compute $G(\mathbf{P}^{k,d})$ for all the possible k and d such that $1 \leq k \leq N/2^d$ and $1 \leq d \leq \log_2 N$ in $O(N)$ time by dynamic programming. Consequently, all of the $F(\mathbf{P}^{k,d})$ values can also be computed in $O(N)$ time. For each d $(1 \leq d \leq \log_2 N)$, let K_d be the sorted list of integers k $(1 \leq k \leq N/2^d)$ according to the $F(\mathbf{P}^{k,d})$ values. K_d can be computed in $O((N \log N)/2^d)$ time. Our preprocessing algorithm in this section computes all these $F(\mathbf{P}^{k,d})$ and K_d for all the k and d, which can be done in $O(N \log N)$ time in total.

By doing a binary search on K_d, we can find all the k such that $x \leq F(\mathbf{P}^{k,d}) \leq y$ for any x, y, and d, in $O(\log(N/2^d) + occ)$ time, where occ is the number of the outputs. Hence, if we are given any structure $\mathbf{S}$ of length 2^d and the value of $F(\mathbf{S})$, we can list all the k such that $D(\mathbf{S}, \mathbf{P}^{k,d}) \leq c$ in $O(\log(N/2^d) + occ)$ time, as $F(\mathbf{S}) - c \leq F(\mathbf{P}^{k,d}) \leq F(\mathbf{S}) + c$ *iff* $D(\mathbf{S}, \mathbf{P}^{k,d}) \leq c$. Let $d_{\mathbf{Q}}$ be the largest d such that, for any i, there exists some k such that $\mathbf{P}^{k,d}$, $\mathbf{P}^{k+1,d}$ and $\mathbf{P}^{k+2,d}$ are substructures of $\mathbf{P}[i..i+m-1]$. Explicitly, $d_{\mathbf{Q}} = \lfloor \log_2(m+1) \rfloor - 2$. Let $w_{\mathbf{Q}} = |\mathbf{P}^{k,d_{\mathbf{Q}}}| = 2^{d_{\mathbf{Q}}}$. Notice that $w_{\mathbf{Q}} > m/8$. Let $I_p^{\mathbf{Q}}$ be a set of integers whose remainder is $p-1$ when divided by $w_{\mathbf{Q}}$ $(1 \leq p \leq w_{\mathbf{Q}})$, *i.e.*, integers representable as $p + j \cdot w_{\mathbf{Q}} + 1$ with some integer j.

Now consider comparing the query $\mathbf{Q}$ and substructures $\mathbf{P}[i..i+m-1]$ such that $i \in I_p^{\mathbf{Q}}$. There are $O(N/w_{\mathbf{Q}}) = O(N/m)$ such substructures. Let $k_{\mathbf{Q}} = \lceil (i-1)/w_{\mathbf{Q}} \rceil + 1$. Then, $\mathbf{P}^{k_Q,d_Q}$, $\mathbf{P}^{k_Q+1,d_Q}$, and $\mathbf{P}^{k_Q+2,d_Q}$ are substructures of $\mathbf{P}[i..i+m-1]$. Let $\mathbf{P}_{\mathbf{Q},i,j} = \mathbf{P}^{k_Q+j-1,d_Q}$ for $j = 1$, 2, and 3. Let $\mathbf{Q}_{p,j} = \mathbf{Q}[p + (j-1) \cdot w_{\mathbf{Q}}..p + j \cdot w_{\mathbf{Q}} - 1]$ for $j = 1$, 2, and 3, and let $D_i^* = \frac{1}{2\sqrt{2}}\{\sum_{j=1}^{3}(D(\mathbf{P}_{\mathbf{Q},i,j}, \mathbf{Q}_{p,j}))^2\}^{1/2}$. Then, D_i^* is also a lower bound of $RMSD(\mathbf{P}[i..i+m-1], \mathbf{Q})$, as shown in the following inequality:

$$D_i^* \leq \frac{1}{2\sqrt{2}}\{\sum_{j=1}^{3}(D(\mathbf{P}_{\mathbf{Q},i,j}, \mathbf{Q}_{p,j}))^2\}^{1/2} < RMSD(\mathbf{P}[i..i+m-1], \mathbf{Q}). \quad (8)$$

Notice that $D(\mathbf{P}_{\mathbf{Q},i,j}, \mathbf{Q}_{p,j}) < 2\sqrt{2}c$ for any j, if $D_i^* < c$. Given the value of $F(\mathbf{Q}_{p,j})$, we can list all $i \in I_p^{\mathbf{Q}}$ such that $D(\mathbf{P}_{\mathbf{Q},i,j}, \mathbf{Q}_{p,j}) < 2\sqrt{2}c$ in $O(\log(N/m) + occ)$ time for any j, by a binary search on the list $K_{d_{\mathbf{Q}}}$, where occ is the number of the i to be listed. Let the list be Y_j $(j = 1, 2, 3)$. Note that $F(\mathbf{Q}_{p,j})$ for all j and p $(1 \leq j \leq 3, 1 \leq p \leq w_{\mathbf{Q}})$ can be computed in $O(m)$ time in total.

Then the same discussion as in section 5.1 can be done. According to the discussions in section 3.3, $\langle |Y_j| \rangle$ is in $O((N/m)/\sqrt{m}) = O(N/m^{1.5})$, under the random-walk assumption on $\mathbf{P}$. The set of the start positions $i \in I_p^{\mathbf{Q}}$ of similar (*i.e.*, the corresponding RMSD is at most c) substructures must be included in all of the three lists: Y_1, Y_2 and Y_3. Thus, the next thing to do is to choose the common positions from the three lists. By using a hashing technique, it can be done in $O(|Y_1| + |Y_2| + |Y_3|)$ time, which is $O(N/m^{1.5})$ under the random-walk assumption. Let Y denote the list of the positions commonly listed in Y_1, Y_2 and Y_3. As $\mathbf{P}_{\mathbf{Q},i,1}$, $\mathbf{P}_{\mathbf{Q},i,2}$ and $\mathbf{P}_{\mathbf{Q},i,3}$ are independent random walks, $\langle |Y| \rangle$ is estimated to be in $O((N/m)/(\sqrt{m})^3) = O(N/m^{2.5})$. For substructures at the positions $i \in Y$, we finally have to compute the RMSD to check whether the actual RMSD is at most c, if $D_i^* < c$. It takes at most $O(m \cdot \langle |Y| \rangle) = O(N/m^{1.5})$ expected time under the random-walk assumption. Thus the total computational time to enumerate all the positions i of similar substructures such that $i \in I_p^{\mathbf{Q}}$ is $O(N/m^{1.5} + \log(N/m))$.

To enumerate all the positions of similar structures, we execute the above for all p $(1 \leq p \leq w_{\mathbf{Q}})$. Thus the total expected query time complexity is $O(m+w_{\mathbf{Q}} \cdot (N/m^{1.5}+\log(N/m))) = O(N/\sqrt{m}+m\log(N/m))$ under the random-walk assumption.

6 Computational Experiments on the PDB Database

We did computational experiments using the whole PDB database [3] of the date September 5th, 2008. Note that more detailed results will be given in the full version of this paper. The database contains 52,821 entries, which include 244,719 chains of proteins. The total number of amino acids of all the chains is 38,267,694. We used the C_α coordinates as the representative coordinates of each amino acid. We did experiments of searching queries of 10 different lengths. In each experiment, we selected 100 substructures of each specified length randomly from the whole database, as sample queries. We used 1 CPU of 1200MHz UltraSPARC III Cu on a SunFire 15K super computer for each experiment.

Table 1 shows the results. Each column shows the result of the queries of each specified length. The '#Substructures' row shows the number of substructures of each specified length in the PDB database. The '#Hits' row shows the average number of hits, *i.e.*, the average number of substructures whose RMSDs to queries are at most 1Å, among the results of the 100 random queries of each specified length. The 'A1', 'A2', 'A3', 'Naive', and 'FFT' rows show the average computation time for the 5 algorithms: A1: the $O(N\sqrt{m})$-time algorithm proposed in section 3, A2: the linear-time algorithm proposed in section 4, A3: another linear-time algorithm that uses the lower bound D_i'' proposed in section 5.1, Naive: the previously known standard $O(Nm)$-time algorithm, and FFT: the previously known $O(N \log N)$-time algorithm based on the FFT (introduced in section 2.3). The algorithms A1,

Table 1. Experimental results over the PDB database for various-length queries

Query length	20	40	60	80	100
#Substructures	33,722,208	29,299,006	25,273,633	21,692,707	18,634,620
#Hits	15,093.64	38.07	27.36	32.90	28.61
A1 (sec)	119.87	98.94	98.94	92.43	85.80
A2 (sec)	117.39	58.86	44.01	36.41	36.25
A3 (sec)	151.52	74.56	33.63	25.54	20.46
Naive (sec)	423.52	447.01	450.39	442.13	428.06
FFT (sec)	551.94	531.92	505.52	463.01	425.77

Query length	120	140	160	180	200
#Substructures	16,134,096	14,084,515	12,362,509	10,884,548	9,559,056
#Hits	27.26	27.71	16.01	17.70	23.21
A1 (sec)	75.58	71.22	59.47	63.48	59.98
A2 (sec)	32.84	30.39	27.27	23.12	25.71
A3 (sec)	17.30	15.85	14.25	12.78	12.91
Naive (sec)	415.24	395.54	378.87	361.43	342.50
FFT (sec)	399.83	367.76	330.57	307.89	293.03

A2 and A3 are all the same algorithms, except for the lower bounds used in them.

In the experiments, we achieved 3.6 to 28 times speed-up against any of the previous algorithms for any-length queries, if we choose to use the lower bound D_i'' when the query is longer than 40, and choose D_i' otherwise. Moreover, the experiments show that our linear-expected-time algorithms run actually in linear time on the PDB database, *i.e.*, the algorithm is not influenced by the difference of query lengths.[4] It means our random-walk assumption is very reasonable for analyses of protein structure databases.

7 Concluding Remarks

We proposed the first linear-expected-time algorithm for searching similar substructures from structure databases, based on the RMSD measure. Moreover, we proposed several preprocessing algorithms that enable theoretically even faster queries. The performance of our algorithms is examined by computational experiments on the whole PDB database.

As for the future work, it would be very interesting to apply our techniques against protein alignment problems that consider insertions and deletions, though it is known to be theoretically much more difficult. Another challenging task would be to design a deterministically linear-time algorithm for our problem. It is also very interesting to extend our techniques against similar problems

[4] Notice that the numbers of substructures of some specified length in the database decreases as the length increases.

in higher dimensions. Moreover, our technique should be applicable to many problems in other research fields such as robotics and computer vision.

Acknowledgement

The author would like to thank Jesper Jansson, Gregory Kucherov, and Kunihiko Sadakane for invaluable discussions. This work was partially supported by the Grant-in-Aid for Young Scientist (B) No. 20700264 from the Ministry of Education, Culture, Sports, Science and Technology of Japan. The author used the super computer system of the 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. Aung, Z., Tan, K.-L.: Rapid retrieval of protein structures from databases. Drug Discovery Today 12, 732–739 (2007)
3. 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)
4. Boyd, R.H., Phillips, P.J.: The Science of Polymer Molecules: An Introduction Concerning the Synthesis. In: Structure and Properties of the Individual Molecules That Constitute Polymeric Materials. Cambridge University Press, Cambridge (1996)
5. Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex Fourier series. Math. Comput. 19, 297–301 (1965)
6. de Gennes, P.-G.: Scaling Concepts in Polymer Physics. Cornell University Press (1979)
7. 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)
8. Eidhammer, I., Jonassen, I., Taylor, W.R.: Structure comparison and structure patterns. J. Comput. Biol. 7(5), 685–716 (2000)
9. Flory, P.J.: Statistical Mechanics of Chain Molecules. Interscience, New York (1969)
10. Gerstein, M.: Integrative database analysis in structural genomics. Nat. Struct. Biol., 960–963 (2000)
11. Golub, G.H., Van Loan, C.F.: Matrix Computation, 3rd edn. John Hopkins University Press (1996)
12. Kabsch, W.: A solution for the best rotation to relate two sets of vectors. Acta Cryst. A32, 922–923 (1976)
13. Kabsch, W.: A discussion of the solution for the best rotation to relate two sets of vectors. Acta Cryst. A34, 827–828 (1978)
14. Kallenberg, O.: Foundations of Modern Probability. Springer, Heidelberg (1997)
15. Kramers, H.A.: The behavior of macromolecules in inhomogeneous flow. J. Chem. Phys. 14(7), 415–424 (1946)

16. 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)
17. 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)
18. Shibuya, T.: Prefix-shuffled geometric suffix tree. In: Ziviani, N., Baeza-Yates, R. (eds.) SPIRE 2007. LNCS, vol. 4726, pp. 300–309. Springer, Heidelberg (2007)
19. Shibuya, T.: Efficient substructure RMSD query algorithms. J. Comput. Biol. 14(9), 1201–1207 (2007)