社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2009－AL－123　(4)
2009／3／5

線形時間タンパク質３次元構造探索アルゴリズム

渋谷 哲朗

東京大学医科学研究所ヒトゲノム解析センター
E-mail: tshibuya@hgc.jp

概要: RMSD(Root Mean Square Deviation) は、２つの同じ長さの座標列を比較するのに最もよく用いられる指標であり、分子生物学、コンピュータグラフィックス、ロボティクス等の様々な分野で用いられている。本論文では、タンパク質などの分子の３次元立体構造データベースから、クエリーとする立体構造からの RMSD が与えられた閾値よりも小さい部分構造をすべて求める、という最も基本的な問題に対し、平均計算時間が線形時間となる初めてのアルゴリズムを提案する。なお、この問題では、これまで平均計算量を考える試みはまったくなされておらず、本論文はその初めての試みでもある。また、このアルゴリズムは理論的にこれまでのアルゴリズムより高速であるだけでなく、実用上も高速なアルゴリズムであり、タンパク質立体構造データベース PDB 上の実験では、従来手法に対し、3.6 倍から 28 倍もの高速化を実現した。また、さらに、平均計算量が高速な検索を実現する索引手法についても提案する。

Linear-Time Algorithm for Searching Protein 3-D Structures

Tetsuo Shibuya

Human Genome Center, Institute of Medical Science, University of Tokyo
E-mail: tshibuya@hgc.jp

Abstract: The RMSD is one of the most fundamental similarity measures for comparing two sets of coordinates. In this paper, we propose a new breakthrough linear-expected-time algorithm for the basic 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. It is not only a theoretically significant improvement over previous algorithms, but also a practically faster algorithm, according to computational experiments. We also propose a series of preprocessing algorithms that enable even faster queries. The experiments show that our algorithm is 3.6 to 28 times faster than the previous algorithms for ordinary queries against the PDB database. The experiments also show that there is consistency between the above theoretical results and the experimental results.

1 Introduction

3-D structure database searching of molecules, especially proteins, plays a more and more important role in molecular biology [2, 9, 11], and 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 deviation) [1, 8, 12, 13, 16, 19] is a fundamental measure to determine the geometric similarity between two same-length sequences of 3-D coordinates. It has been widely used for a long time, not only for structure comparison of proteins or other molecules, 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 for more details.) 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 around 2Å.

Our results

The best-known 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, whether it is the worst-case analysis or the expected-time analysis (see section 2.3). We propose the first linear-expected-time (*i.e.*, $O(N)$) algorithm. Note that we assume that the structures of the chain molecules in the database can be considered as random walks in 3-D space in the analysis of its expected time. This assumption is often used to analyze properties of chain molecules [4, 7, 10, 15] (see section 2.4). We also examine the validness of the analysis through computational experiments.

We 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 linear-expected-time algorithm by computational experiments on the whole PDB database. In the experiments, no inconsistency is observed between the above theoretical results and the experimental results: The computation time of our $O(N)$ algorithm is not influenced by changes of query lengths, in contrast to previously known algorithms. It means that our random-walk assumption is very reasonable for analyses of protein structures. Moreover, our linear-time algorithm is much faster than previous algorithms, *i.e.*, 3.6 to 28 times faster to search for substructures whose RMSDs is at most 1Å. Furthermore, it is always more than 20 times faster than previous algorithms for queries no shorter than 100 aa.

2 Preliminaries

2.1 Notations and Definitions

A chain molecule is represented like $\mathbf{S} = \{\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_n\}$, where $\vec{s}_i$ denotes the 3-D coordinates of the i-th atom. The length n of $\mathbf{S}$ is denoted by $|\mathbf{S}|$. A structure $\mathbf{S}[i..j] = \{\vec{s}_i, \vec{s}_{i+1}, \ldots, \vec{s}_j\}$ $(1 \le i \le j \le n)$ is called a substructure of $\mathbf{S}$. $R \cdot \mathbf{S}$ denotes the structure $\mathbf{S}$ rotated by the rotation matrix R, *i.e.*, $R \cdot \mathbf{S} = \{R\vec{s}_1, R\vec{s}_2, \ldots, R\vec{s}_n\}$. $|\vec{v}|$ denotes the norm of the vector $\vec{v}$. $\vec{0}$ 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 $\mathbf{P} = \{\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_N\}$, and we let $\mathbf{Q} = \{\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_m\}$ 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 structure 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, 8, 12, 13, 16, 19] between two 3-D coordinate sequences $\mathbf{S} = \{\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_n\}$ and $\mathbf{T} = \{\vec{t}_1, \vec{t}_2, \ldots, \vec{t}_n\}$ is defined as the minimum value of $E_{R,\vec{v}}(\mathbf{S}, \mathbf{T}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} |\vec{s}_i - (R \cdot \vec{t}_i + \vec{v})|^2}$ over all the possible rotation matrices R and translation vectors $\vec{v}$. Let $RMSD(\mathbf{S}, \mathbf{T})$ denote the minimum value, and let $\hat{R}(\mathbf{S}, \mathbf{T})$ and $\hat{\vec{v}}(\mathbf{S}, \mathbf{T})$ denote the rotation matrix and the translation vector that minimizes $E_{R,\vec{v}}(\mathbf{S}, \mathbf{T})$. We can compute $RMSD(\mathbf{S}, \mathbf{T})$, $\hat{R}(\mathbf{S}, \mathbf{T})$ and $\hat{\vec{v}}(\mathbf{S}, \mathbf{T})$ in linear time using the singular value decomposition [12, 13].

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. Thus 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, 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 Conformations

The *random-walk model* for chain molecule conformations is a simple but useful model for analyzing their behavior [4, 7, 10, 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.

Consider a chain molecule $\mathbf{S} = \{\vec{s}_0, \vec{s}_2, \ldots, \vec{s}_n\}$ 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.*, $\vec{b}_i = \vec{s}_{i+1} - \vec{s}_i$, is considered as a random vector that satisfies $|\vec{b}_i| = \ell$, and $\vec{b}_i$ is independent from $\vec{b}_j$ for any i and j $(i \neq j)$. If n is large enough, the distribution of the end-to-end vector $\vec{s}_n - \vec{s}_0$ is known to converge to the Gaussian distribution in 3-D space, in which $\langle \vec{s}_n - \vec{s}_0 \rangle = \vec{0}$ and $\langle |\vec{s}_n - \vec{s}_0|^2 \rangle = n \cdot \ell^2$. In the distribution, the probability (or probability density) $W_{n,\ell}(x, y, z)dxdydz$ that $\vec{s}_n - \vec{s}_0$ is located at some position (x, y, z) is:

$$\left(\frac{3}{2\pi n \ell^2}\right)^{\frac{3}{2}} e^{-3(x^2+y^2+z^2)/2n\ell^2} dxdydz. \qquad (1)$$

The random-walk model is known to reflect the behavior of real molecules very well [4], though it ignores many physical/chemical constraints. Even if we assume more complicated models, the behavior of chain molecules does not differ so much [6, 10]. For example, Daynatis and Palierne [6] showed through Monte Carlo simulation that the above end-to-end vector still follows a Gaussian-like distribution, under the so-called self-avoiding random-walk model. Hence, we consider it reasonable to assume the structures in the databases follow the random-walk model, as we do in this paper. Moreover, we will show in section 6 that our experimental results on the PDB database show high consistency with the random-walk model.

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

3.1 An Efficiently-Computable Lower Bound for the RMSD

In this section, we propose a nontrivial, but easily-computable lower bound for the RMSD between any two structures with the same length. Let $\mathbf{U}^{left}$ denote $\{\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_{\lfloor k/2 \rfloor}\}$ and $\mathbf{U}^{right}$ denote $\{\vec{u}_{\lfloor k/2 \rfloor+1}, \vec{u}_{\lfloor k/2 \rfloor+2}, \ldots, \vec{u}_{2\cdot\lfloor k/2 \rfloor}\}$ for a structure $\mathbf{U} = \{\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_k\}$. Let $G(\mathbf{U})$ denote the centroid (center of mass) of the structure $\mathbf{U}$, *i.e.*, $G(\mathbf{U}) = \frac{1}{k}\sum_{i=1}^{k} \vec{u}_i$. 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 $\vec{s}'_i = \vec{s}_i - G(\mathbf{S})$, and $\vec{t}'_i = \vec{t}_i - G(\mathbf{T})$. 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:

$$\begin{aligned}
&RMSD(\mathbf{S},\mathbf{T})\\
\geq\ & \frac{1}{n}\sum_{i=1}^{n}|\vec{s_i'} - \hat{R}(\mathbf{S},\mathbf{T})\cdot\vec{t_i'}|\\
\geq\ & \frac{1}{n}|\sum_{i=1}^{n/2}\{\vec{s_i'} - \hat{R}(\mathbf{S},\mathbf{T})\cdot\vec{t_i'}\}|\\
&+\frac{1}{n}|\sum_{i=n/2+1}^{n}\{\vec{s_i'} - \hat{R}(\mathbf{S},\mathbf{T})\cdot\vec{t_i'}\}|\\
=\ & |G(\mathbf{S}^{left}) - G(\mathbf{S}) - \hat{R}(\mathbf{S},\mathbf{T})\\
&\qquad\cdot\{G(\mathbf{T}^{left}) - G(\mathbf{T})\}|/2\\
&+|G(\mathbf{S}^{right}) - G(\mathbf{S}) - \hat{R}(\mathbf{S},\mathbf{T})\\
&\qquad\cdot\{G(\mathbf{T}^{right}) - G(\mathbf{T})\}|/2\\
=\ & |G(\mathbf{S}^{left}) - G(\mathbf{S}^{right}) - \hat{R}(\mathbf{S},\mathbf{T})\\
&\qquad\cdot\{G(\mathbf{T}^{left}) - G(\mathbf{T}^{right})\}|/2\\
\geq\ & |\{|G(\mathbf{S}^{left}) - G(\mathbf{S}^{right})|\\
-\ & |G(\mathbf{T}^{left}) - G(\mathbf{T}^{right})|\}|/2\\
=\ & D(\mathbf{S},\mathbf{T}). \qquad (2)
\end{aligned}$$

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

$$\begin{aligned}
RMSD(\mathbf{S},\mathbf{T}) &\geq \sqrt{\frac{n-1}{n}}RMSD(\mathbf{S}^-,\mathbf{T}^-)\\
&\geq \sqrt{\frac{n-1}{n}}D(\mathbf{S}^-,\mathbf{T}^-) = D(\mathbf{S},\mathbf{T}), \quad (3)
\end{aligned}$$

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

To solve our problem presented in section 1, we have to check the RMSD between the query structure $\mathbf{Q}$ and each of the substructures $\mathbf{P}[i..i+m-1]$ in the database. If the above lower bound can be computed very efficiently, we may use the value to filter 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 $G(\mathbf{P}[i..i+\lfloor m/2\rfloor-1] = G(\mathbf{P}[i-1..i+\lfloor m/2\rfloor-2]) - \frac{1}{\lfloor m/2\rfloor}(\vec{p}_{i-1}-\vec{p}_{i+\lfloor m/2\rfloor-1})$. 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 $D_i = D(\mathbf{P}[i..i+m-1],\mathbf{Q})$ for all $i$ such that $1\leq i\leq N-m+1$.
2  for (all $i$ such that $1\leq i\leq N-m+1$) {
3     if ($D_i\leq c$) {
4        if ($RMSD(\mathbf{P}[i..i+m-1],\mathbf{Q})\leq c$)
5              { Output $i$ as a position of a substructure similar to the query. }
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 $O(N/\sqrt{m})$ and consequently the expected time complexity of the above algorithm is $O(N\sqrt{m})$, under the random-walk assumption.

3.3 Computational Time Analysis

Consider a structure $\mathbf{S} = \{\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_{2n}\}$ 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 $\vec{s}_i$ can be represented as $\vec{s}_1 + \sum_{j=1}^{i-1}\vec{b}_j$, where $\vec{b}_i$ is an independent random vector that satisfies $|\vec{b}_i| = 1$. 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:

$$\begin{aligned}
H(\mathbf{S}) &= \frac{1}{n}\sum_{i=1}^{n}(\vec{s}_1 + \sum_{j=1}^{i-1}\vec{b}_j)\\
&\quad -\frac{1}{n}\sum_{i=n+1}^{2n}(\vec{s}_1 + \sum_{j=1}^{i-1}\vec{b}_j)\\
&= -\{\sum_{i=1}^{n}\frac{i}{n}\cdot\vec{b}_i + \sum_{i=n+1}^{2n-1}\frac{2n-i}{n}\cdot\vec{b}_i\}. \quad (4)
\end{aligned}$$

Let $\vec{b_i'}$ denote $\frac{i}{n}\cdot\vec{b}_i$ if $i\leq n$ and $\frac{2n-i}{n}\cdot\vec{b}_i$ if $i > n$. Then $H(\mathbf{S})$ can be described as $\sum_{i=1}^{2n}\vec{b_i'}$.

Let z_i denote the z coordinate of $\vec{b}_i$ and z'_i denote the z coordinate of $\vec{b}'_i$. It is easy to see that $\langle z_i \rangle = 0$ and $var(z_i) = 1/3$, as $\vec{b}_i$ is a random vector that satisfies $|\vec{b}_i| = 1$. Let M_n be $\sum_{i=1}^{2n} \langle |z'_i - \langle z'_i \rangle| \rangle^{2+\delta} / \sqrt{\sum_{i=1}^{2n} var(z'_i)}^{2+\delta}$ 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:

$$\begin{aligned} M_n &\le \sum_{i=1}^{2n} \langle |z'_i| \rangle^2 / \sqrt{\sum_{i=1}^{2n} \langle |z'_i| \rangle^2}^{2+\delta} \\ &= \{\sum_{i=1}^{2n} \langle |z'_i| \rangle^2\}^{-\delta/2} \\ &= \{\frac{1}{3n^2} \cdot \{\sum_{i=1}^{n} i^2 + \sum_{i=n+1}^{2n-1} (2n-i)^2\}\}^{-\delta/2} \\ &= \{\frac{2}{9}n + \frac{1}{9n}\}^{-\delta/2} \to 0 \quad (n \to \infty). (5) \end{aligned}$$

Hence, we conclude that $\sum_i^{2n} z'_i$ converges to the Gaussian distribution. It also means $H(\mathbf{S})$ 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 $H(\mathbf{S})$ can be computed as follows:

$$\begin{aligned} var(H(\mathbf{S})) &= \langle |H(\mathbf{S})|^2 \rangle - \langle |H(\mathbf{S})| \rangle^2 \\ &= \langle |H(\mathbf{S})|^2 \rangle \\ &= \langle | \sum_{i=1}^{n} \frac{i}{n} \cdot \vec{b}_i + \sum_{i=n+1}^{2n-1} \frac{2n-i}{n} \cdot \vec{b}_i |^2 \rangle \\ &= \frac{1}{n^2} \{\sum_{i=1}^{n} i^2 + \sum_{i=n+1}^{2n-1} (2n-i)^2\} \\ &= \frac{2}{3}n + \frac{1}{3n} \approx \frac{2}{3}n, \end{aligned} \tag{6}$$

as $\langle \vec{b}_i \cdot \vec{b}_j \rangle = 0$ if $i \neq j$. Moreover, it is easy to see that $\langle H(\mathbf{S}) \rangle = \vec{0}$. Thus the distribution of $H(\mathbf{S})$ is the same as the distribution of random walks of length $2n/3$. Hence the probability distribution $Z_n(x,y,z)dxdydz$ of $H(\mathbf{S})$ is $(\frac{9}{4\pi n})^{\frac{3}{2}} e^{-9(x^2+y^2+z^2)/4n} dxdydz$. Consequently, the probability (or probability density) that $|H(\mathbf{S})| = r$ is:

$$Z_n(r)dr = 4\pi r^2 (\frac{9}{4\pi n})^{\frac{3}{2}} e^{-9r^2/4n} dr. \tag{7}$$

Integrating $Z_n(r)dr$, we obtain $Prob(x \le |H(\mathbf{S})| \le y) = \int_{r=x}^{y} Z_n(r)dr$. $Z_n(r)$ takes the maximum value at $r_{max} = \frac{2}{3}\sqrt{n}$ and $Z_n(r_{max}) = 6e^{-1}/\sqrt{\pi n}$. Thus $Prob(x \le |H(\mathbf{S})| \le y)$ 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 $Prob(|D(\mathbf{S},\mathbf{T})| \le c) = Prob(F(\mathbf{T}) - c \le F(\mathbf{S}) \le F(\mathbf{T}) + c) = Prob(2 \cdot F(\mathbf{T}) - 2c \le |H(\mathbf{S})| \le 2 \cdot F(\mathbf{T}) + 2c))$ 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 $Prob(D_i \le c)$ in the line 3 of the algorithm in section 3.2 is in $O(1/\sqrt{m})$ 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. 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

In this section, we improve the expected time complexity of the algorithm 1 by using a different lower bound for the RMSD. From the definition of the RMSD, we can deduce that

$$\begin{aligned} &RMSD(\mathbf{S},\mathbf{T}) \\ &\ge \sqrt{t} \cdot \{(RMSD(\mathbf{S}^{left},\mathbf{T}^{left}))^2 \\ &\quad +(RMSD(\mathbf{S}^{right},\mathbf{T}^{right}))^2\}^{1/2} \\ &\ge \sqrt{t} \cdot \{(D(\mathbf{S}^{left},\mathbf{T}^{left}))^2 \\ &\quad +(D(\mathbf{S}^{right},\mathbf{T}^{right}))^2\}^{1/2}, \end{aligned} \tag{8}$$

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 (8) 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 $1 \le i \le N - m + 1$,
        compute $D'_i = \{(D^{left}(\mathbf{P}[i..i+m-1],\mathbf{Q}))^2 + (D^{right}(\mathbf{P}[i..i+m-1],\mathbf{Q}))^2\}^{1/2}$.
2   for (all $i$ such that $1 \le i \le N - m + 1$) {
3       if ($D'_i \le c$) {
4           if ($RMSD(\mathbf{P}[i..i+m-1],\mathbf{Q}) \le c$)
5               { Output $i$ as a position of a substructure similar to the query. }
6      }
7   }
```

The only difference from the Algorithm 1 is the lower bound D'_i computed in the line 1. Note that there is no difference in the time complexity of the line 1, *i.e.*, $O(N)$.

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 both in $O(1/\sqrt{m})$, under the assumption that the structure $\mathbf{P}$ follows the random-walk model. Moreover, the two probabilities are independent from each other. 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

In this section, 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.

Now we consider yet another lower bound for the RMSD, as follows:

$$
\begin{aligned}
&RMSD(\mathbf{P}[i..i+m-1],\mathbf{Q})\\
&\geq \sqrt{\frac{m'}{m}}\cdot\{\sum_{j=1}^{3}(RMSD(\mathbf{P}[i+(j-1)\cdot m'..\\
&\qquad\qquad i+j\cdot m'-1],\\
&\qquad \mathbf{Q}[1+(j-1)\cdot m'..j\cdot m']))^2\}^{1/2}\\
&\geq \sqrt{\frac{m'}{m}}\cdot\{\sum_{j=1}^{3}(D(\mathbf{P}[i+(j-1)\cdot m'..\\
&\qquad\qquad i+j\cdot m'-1],\\
&\qquad \mathbf{Q}[1+(j-1)\cdot m'..j\cdot m']))^2\}^{1/2}. \quad (9)
\end{aligned}
$$

Let D''_i be the lower bound given in expression (9). 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 $1 \leq j \leq 3$). According to the previous discussions, 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, according to the discussion in section 3.1. 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 $O(m + N/\sqrt{m} + m\cdot N/m^{1.5} + \log N) = O(m + N/\sqrt{m})$, as we can ignore the term '$\log N$'.

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 (9) 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 the algorithm requires $O(N \log N)$ space to store all the L_w, which might be undesired for huge databases. In the next section, we will propose another preprocessing algorithm that uses only $O(N)$ space for queries of arbitrary lengths.

5.3 Preprocessing with Linear 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. Then, as $w_{\mathbf{Q}} > m/8$, the following inequality holds:

$$\begin{aligned} & RMSD(\mathbf{P}[i..i+m-1], \mathbf{Q}) \\ & > \frac{1}{2\sqrt{2}}\{\sum_{j=1}^{3}(RMSD(\mathbf{P}_{\mathbf{Q},i,j}, \mathbf{Q}_{p,j}))^2\}^{1/2} \\ & \geq \frac{1}{2\sqrt{2}}\{\sum_{j=1}^{3}(D(\mathbf{P}_{\mathbf{Q},i,j}, \mathbf{Q}_{p,j}))^2\}^{1/2}. \end{aligned} \quad (10)$$

Let D_i'' be the lower bound given in expression (10). 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, to examine the performance of our linear-time algorithm. 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. In the following experiments, we used the SunFire 15K super computer with 96 CPUs of 1200MHz UltraSPARC III Cu and 288 GB memory. Note that we used only 1 CPU for each experiment. In the experiments, we searched for all the substructures in the PDB database such that the RMSD to the query is at most 1Å, for queries of various lengths.

The experiments show that our linear-expected-time algorithm proposed in section 4 is actually linear-time algorithm on the PDB database, *i.e.*, the algorithm is not influenced by the difference of query lengths. For example, the average search time over the PDB database for queries of length 40 is 58.86 seconds. It is 36.25 seconds for queries of length 100, and is 25.71 seconds for queries of length 200, all of which are about 2 seconds per 1,000,000 substructures in the PDB database. The fact that the query time is not influenced by the query length means our random-walk assumption is very reasonable for analyses of protein structure databases.

Moreover, our linear-expected-time algorithm proposed in section 4 is about 3.6 to 13.3 times faster than any of the previous algorithms. For example, our algorithm can search for queries of length 100 in 36.25 seconds, which is 11.7 times faster than previous algorithms: The naive $O(Nm)$ algorithm can do the same in 428.06 seconds, and the FFT-based $O(N\log m)$ algorithm can do the same in 425.77 seconds. In cases that queries are longer, our algorithm can be improved further by using the lower bound D''_i proposed in section 5.1 instead of D'_i used in the algorithm 2. For example, we can search for structures similar to queries of length 100 only in 20.46 seconds, which is about 20 times faster than the two previously known algorithms. If we choose to use the lower bound D''_i when the query is longer than 40, and choose D'_i otherwise, we achieve 3.6 to 28 times speed-up against any of the previous algorithms for any-length queries.

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 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] K. S. Arun, *et al. IEEE Trans Pat. Anal. Machine Intell.*, 9, 698–700, 1987.

[2] Z. Aung *et al. Drug Disc. Today*, 12, 732–739, 2007.

[3] H. M. Berman, *et al. Nucl. Acids Res.*, 28, 235–242, 2000.

[4] R. H. Boyd, *et al. The Science of Polymer Molecules: An Introduction Concerning the Synthesis, Structure and Properties of the Individual Molecules That Constitute Polymeric Materials*, Cambridge University Press, 1996.

[5] J. W. Cooley, *et al. Math. Comput.*, 19, 297–301, 1965.

[6] J. Dayantis, *et al. J. Chem. Phys.*, 95, 6088–6099, 1991.

[7] P.-G. de Gennes. *Scaling Concepts in Polymer Physics*, Cornell University Press, 1979.

[8] D. W. Eggert, *et al. Mach. Vis. and Appl.*, 9, 272–290, 1997.

[9] I. Eidhammer, *et al. J. Comput. Biology*, 7(5), 685–716, 2000.

[10] P. J. Flory. *Statistical Mechanics of Chain Molecules*, Interscience, New York, 1969.

[11] M. Gerstein. *Nat. Struct. Biol.*, Suppl., 960–963, 2000.

[12] W. Kabsch. *Acta Cryst.*, A32, 922–923, 1976.

[13] W. Kabsch. *Acta Cryst.*, A34, 827–828, 1978.

[14] O. Kallenberg. *Foundations of Modern Probability*, Springer-Verlag, 1997.

[15] H. A. Kramers. *J. Chem. Phys.*, 14(7), 415–424, 1946.

[16] J. T. Schwartz, *et al. Intl. J. of Robotics Res.*, 6, 29–44, 1987.

[17] T. Shibuya. *LNCS 4009*, 84–93, 2006.

[18] T. Shibuya. *LNCS 4726*, 300–309, 2007.

[19] T. Shibuya. *J. Comput. Biol.*, 14(9), 1201–1207, 2007.