Proceedings of the 28th Annual Hawaii International Conference on System Sciences — 1995

New Hashing Techniques and Their Application to a Protein Structure Database System

Tatsuya Akutsu
Computer Science Department
Gunma University
Kiryu, Gunma 376, Japan
akutsu@cs.gunma-u.ac.jp

Kentaro Onizuka
Human Interface Research Laboratory
Matsushita Research Institute Tokyo, Inc.
Higashimita, Tama-Ku, Kawasaki 214, Japan
onizuka@mrit.mei.co.jp

Masato Ishikawa
Institute for New Generation
Computer Technology
Mita, Minato-Ku, Tokyo 108, Japan
ishikawa@icot.or.jp

Abstract

We have devised novel methods to evaluate the structural similarity of proteins. In this paper, we compare them. In each method, a hash vector is associated with each fixed-length fragment of three-dimensional protein structure. Then, we analyze the similarity between fragments by evaluating the difference between two hash vectors. The novel aspect of the methods is that the following property is proved theoretically: if the root mean square deviation between two fragments is small, then the distance between the hash vectors associated with the fragments is small. The methods were compared with the previous methods using PDB data, and were shown to be much faster. One of the new hashing methods is already included in PROTEIX, a database management system for protein structures. The features of PROTEIX are also described in this paper.

1 Introduction

Currently, the number of proteins, for which three-dimensional (3D) structures are known, is about 1000 and it grows year by year. Thus, a database management system for 3D protein structures is required for studying structural relations among a large number of protein structures, which may contribute to molecular biology and drug design.

3D data of proteins are collected in Brookhaven's Protein Data Bank (PDB) as a set of text files [3]. A number of tools which can treat PDB files have been developed. Although quick searching for similar structures seems very important for interactive uses of tools, few tools have been developed that allow such a search. Thus, we have developed novel methods for quick substructure searching, by which similar fragments can be retrieved from a database within a few seconds on a standard UNIX workstation. One of the methods has been adapted to PROTEIX, which is a database management system for 3D protein structures being developed by us. This paper describes the methods as well as the PROTEIX system.

Here, we briefly review previous work. While several problems can be considered for searching similar structures, this paper focuses on the following problem (the *substructure search problem*): given a fragment structure P and a positive real number δ, find all proteins in a database each of which contains at least a fragment R such that the root mean square deviation (*rmsd*, in short) between P and R does not exceed δ. A large number of pattern matching algorithms for 3D protein structures have been proposed [2, 8, 9, 11, 12, 15, 16, 17]. However, few of them can be used for quick substructure searching. Nussinov and Wolfson's method [8] based on the geometric hashing technique may be used. However, it requires long preprocessing time and large memory space. Other types of geometric hashing techniques have also been proposed in computer vision [6, 7], but none of them seems to be suited to this problem. An FFT(Fast Fourier Transform)-based algorithm, which was developed for computer vision by Schwartz and Sharir [13], may be used for quick substructure searching. Although their algorithm is elegant and efficient, it is not practical for typical sizes of fragment structures. Indeed, experimental results show that it is better than a naive method only when the number of residues is more than $200 \sim 300$ [14].

In the previous paper, one of the authors proposed a method for quick substructure searching, named the *least-squares hashing method* [1]. It is quite different from the geometric hashing technique used by Nussinov and Wolfson [8]. It is rather similar to the conventional hashing technique, which is widely used in database management systems. In the conventional hashing technique, an integer number is computed for each object so that the numbers are equal if the objects are identical. In the least-squares hashing method, this technique is modified for protein structures as follows. Since what should be searched for are not identical substructures but similar substructures, a vector of real numbers is associated with each fixed length fragment of protein structure. Moreover, the following property is required for hash vectors: if *rmsd* between two fragments is small, the distance between the associated hash vectors is small. In the least-squares hashing method, the above property is satisfied in most cases. However, it is not proved theoretically. Indeed, the least-squares hashing method sometimes fails to find similar substructures. It is a crucial weak point of the least-squares hashing method. Thus, we have developed new hashing methods, in which the above property is theoretically proved. As far as we know, the proposed hash vectors are the first ones for which the above property is theoretically proved. Moreover, the new methods are much faster than the least-squares hashing method.

The organization of this paper is as follows. First, the *rmsd* fitting is reviewed, and the substructure search problem is defined formally using *rmsd*. Next, new hashing methods and their theoretical properties are described in Section 3, and experimental results for comparing them are described in Section 4. A database management system PROTEIX is then overviewed. Finally, we conclude with a brief summary and future work.

2 Preliminaries

First, we consider representation of 3D protein structures. As we are only interested in representing an outline of 3D structure, we follow the common procedure of ignoring side chains and consider only Cα atoms (or the carbon and nitrogen atoms in the main chain), which are treated as points in 3D space. Only the geometry of protein structures is considered and details such as the identity of specific atoms are ignored. Thus, each protein structure is treated as a sequence of points. For each structure $P = (\boldsymbol{p}^1, \cdots, \boldsymbol{p}^n)$, $P_{i,j}$ denotes the fragment $(\boldsymbol{p}^i, \boldsymbol{p}^{i+1}, \cdots, \boldsymbol{p}^j)$ of P.

2.1 Root mean square deviation

Here, we briefly review the root mean square deviation, which is used as a common measure for comparing two protein structures in molecular biology. *Rmsd* fitting is a kind of least-squares fitting method for two sequences of points, and was developed by several persons independently [5, 10, 13].

Let $P = (\boldsymbol{p}^1, \cdots, \boldsymbol{p}^n)$ and $Q = (\boldsymbol{q}^1, \cdots, \boldsymbol{q}^n)$ be two sequences of points. We assume that P is translated so that its centroid $(\frac{1}{n}\sum_{k=1}^{n}\boldsymbol{p}^k)$ is at the origin. We also assume that Q is translated in the same way. For each point or vector $\boldsymbol{s}$, $\boldsymbol{s}_i$ $(i = 1, 2, 3)$ denotes the i-th (X,Y,Z) coordinate value of $\boldsymbol{s}$. Let

$$d(P, Q, R, \boldsymbol{a}) = \sqrt{\frac{1}{n}\sum_{k=1}^{n}\|R\boldsymbol{p}^k + \boldsymbol{a} - \boldsymbol{q}^k\|^2}\ ,$$

where R is a rotation matrix, $\boldsymbol{a}$ is a translation vector, and $\|\boldsymbol{x}\|$ denotes the length of a vector $\boldsymbol{x}$. Then, the *rmsd* value $d(P, Q)$ between P and Q is defined by $d(P, Q) = \min_{R,\boldsymbol{a}} d(P, Q, R, \boldsymbol{a})$.

$d(P, Q, R, \boldsymbol{a})$ is minimized when $\boldsymbol{a}$ is the zero vector and $R = (A^tA)^{1/2}A^{-1}$, where the matrix $A = (A_{ij})$ $(i, j = 1, 2, 3)$ is given by $A_{ij} = \sum_{k=1}^{n}\boldsymbol{p}_i^k\boldsymbol{q}_j^k$, and $A^{1/2} = B$ means $BB = A$ [13]. Thus, $d(P, Q)$, R and $\boldsymbol{a}$ can be computed in $O(n)$ time, where $O(f(n))$ time means that the computation time is at most $C \times f(n)$ for some constant C.

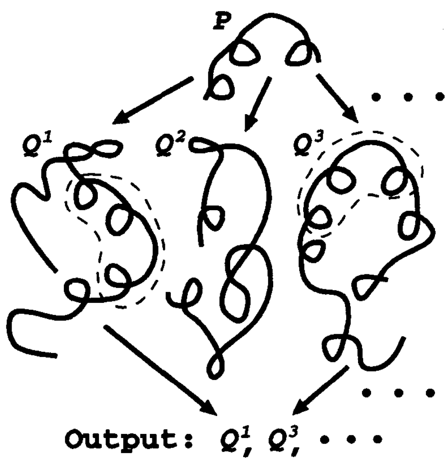

Fig. 1. Substructure search problem.

2.2 Substructure search

Using *rmsd*, we define the substructure search problem as follows (see Fig. 1):

Input: A (pattern) fragment $P = (\boldsymbol{p}^1, \cdots, \boldsymbol{p}^m)$, a real number $\delta > 0$ and a set of proteins $QS = \{Q^1, \cdots, Q^N\}$,

Output: All structures Q^j each of which contains at least a fragment $Q^j_{i,i+m-1}$ such that $d(P, Q^j_{i,i+m-1}) \leq \delta$.

The substructure search problem can be solved by a naive method which computes $rmsd$ for all $Q^j_{i,i+m-1}$'s. However, it takes $O(Nmn)$ time, where we assume that the length of each Q^j is $O(n)$. In fact, experimental results described in Section 4 show that it takes about a minute. It is too long for interactive uses of database management systems.

3 New hash vectors

3.1 Conditions for hash vectors

Before describing new hash vectors, we describe general conditions which should be satisfied by any hash vector. In conventional hashing methods, an integer number is associated with each object. However, in the hashing methods for protein structures, a vector of reals is associated with each fragment of fixed length. For each fragment $P = (\boldsymbol{p}^1, \cdots, \boldsymbol{p}^H)$ of length H, a hash vector $\boldsymbol{hs}(P)$ is associated. Then, the following conditions should be satisfied by $\boldsymbol{hs}(P)$:

(A) $\boldsymbol{hs}(P)$ is invariant with any isometric transformation (rotation/translation) for P,

(B) $\boldsymbol{hs}(P)$ is close to $\boldsymbol{hs}(Q)$ if $d(P, Q)$ is small.

Although condition (A) may be implied by condition (B), we describe them separately to make the presentation clear. Note that once such a vector is given, $d(P, Q)$ must be computed only when $\boldsymbol{hs}(P)$ is close to $\boldsymbol{hs}(Q)$. In the least-squares hashing method previously proposed [1], condition (A) is satisfied but condition (B) is not necessarily satisfied. Indeed, it sometimes fails to find similar fragments.

3.2 Hash vector - a basic one

Here, we describe new hash vectors. All vectors are computed in a similar way. First, we describe a basic one, denoted by HASH(A).

HASH(A):
$\boldsymbol{hs}(P) = (c_1(P), s_1(P), \cdots, c_D(P), s_D(P))$, where

$$c_i(P) = \alpha \sum_{k=1}^{H} \|\boldsymbol{p}^k - \mathbf{c}\| (\cos(\frac{2\pi i(k-1)}{H}) + \beta),$$
$$s_i(P) = \alpha \sum_{k=1}^{H} \|\boldsymbol{p}^k - \mathbf{c}\| (\sin(\frac{2\pi i(k-1)}{H}) + \beta).$$

Note that $\mathbf{c}$ denotes the centroid of P (i.e., $\mathbf{c} = \sum_{k=1}^{H} \boldsymbol{p}^k$). Thus, $\|\boldsymbol{p}^k - \mathbf{c}\|$ denotes the length between the centroid and the k-th point. Also note that α ($\alpha > 0$) and β ($\beta \geq 0$) are fixed reals and D is a fixed integer, which are to be determined later. Thus, $\boldsymbol{hs}(P)$ is computed in the following way.

(1) Compute the centroid $\mathbf{c}$ of P.
(2) For $i = 1$ to D, repeat (3)-(6).
(3) $c_i \leftarrow 0$, $s_i \leftarrow 0$.
(4) For $k = 1$ to H, repeat (5)(6).
(5) $c_i \leftarrow c_i + \alpha\|\boldsymbol{p}^k - \mathbf{c}\|(\cos(\frac{2\pi i(k-1)}{H}) + \beta)$.
(6) $s_i \leftarrow s_i + \alpha\|\boldsymbol{p}^k - \mathbf{c}\|(\sin(\frac{2\pi i(k-1)}{H}) + \beta)$.
(7) $\boldsymbol{hs}(P) \leftarrow (c_1, s_1, \cdots, c_D, s_D)$.

Note that $\boldsymbol{hs}(P)$ is similar to (a low frequency part of) the Fourier spectrum of the distances between the points and the centroid (see Fig. 2). Although the Fourier spectrum has been already applied to geometric hashing [6], $\boldsymbol{hs}(P)$ is quite different from it.

Now, we will prove that $\boldsymbol{hs}(P)$ defined in HASH(A) satisfies conditions (A) and (B). Condition (A) is trivially satisfied since $\boldsymbol{hs}(P)$ is computed only from the distances between the points and the centroid. To show that condition (B) is satisfied, we first prove the following lemma. Readers who are not interested in details of the theoretical analysis might skip lemmas, proofs and theorems.

Lemma 1: Assume that $P = (\boldsymbol{p}^1, \cdots, \boldsymbol{p}^H)$ and $Q = (\boldsymbol{q}^1, \cdots, \boldsymbol{q}^H)$ are translated so that the centroids are at the origin. Then, $|\sum_{i=1}^{H} \|\boldsymbol{p}^i\| - \sum_{i=1}^{H} \|\boldsymbol{q}^i\|| \leq H\, d(P, Q)$.

Proof: Let $\hat{Q} = (\hat{\boldsymbol{q}}^1, \cdots, \hat{\boldsymbol{q}}^H)$ denote the rotated sequence of Q such that $d(P, \hat{Q}, I, \mathbf{o}) = d(P, Q)$, where $\mathbf{o}$ denotes the zero vector and I denotes the identity matrix.

Then, the following inequality holds:

$$\begin{aligned} |\sum_{i=1}^{H} \|\boldsymbol{p}^i\| - \sum_{i=1}^{H} \|\boldsymbol{q}^i\|| &= |\sum_{i=1}^{H} \|\boldsymbol{p}^i\| - \sum_{i=1}^{H} \|\hat{\boldsymbol{q}}^i\|| \\ \leq \sum_{i=1}^{H} | \|\boldsymbol{p}^i\| - \|\hat{\boldsymbol{q}}^i\| | &\leq \sum_{i=1}^{H} \|\boldsymbol{p}^i - \hat{\boldsymbol{q}}^i\|, \end{aligned}$$

Proceedings of the 28th Hawaii International Conference on System Sciences (HICSS '95)
1060-3425/95 $10.00 © 1995 IEEE

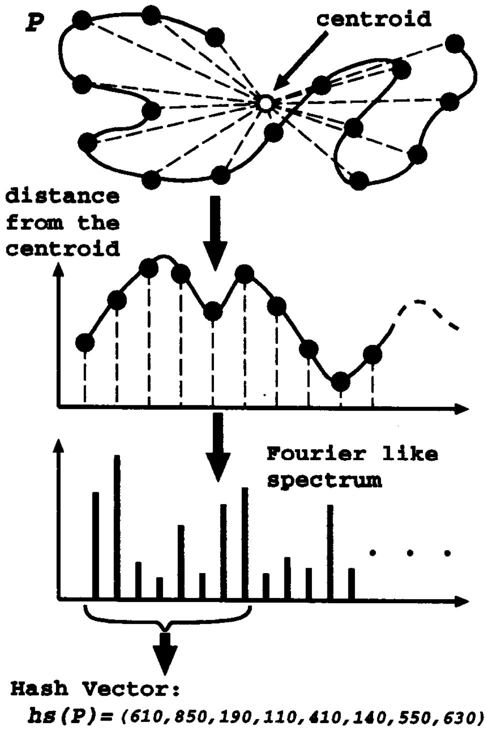

Fig. 2. Computation of a hash vector.

where the last inequality comes from the triangular inequality. Since $t_1 + \cdots + t_H \le \sqrt{H}\sqrt{t_1^2 + \cdots + t_H^2}$ holds for all $t_1 \ge 0, \cdots, t_H \ge 0$,

$$\sum_{i=1}^{H} \|\boldsymbol{p}^i - \hat{\boldsymbol{q}}^i\| \le \sqrt{H}\sqrt{\sum_{i=1}^{H} \|\boldsymbol{p}^i - \hat{\boldsymbol{q}}^i\|^2} = H\, d(P,Q)$$

holds and the theorem follows. □

From Lemma 1, the following theorem is immediately proved, which shows that HASH(A) satisfies condition (B).

Theorem 1: For all i, $|c_i(P) - c_i(Q)| \le H\alpha(1+\beta)d(P,Q)$ and $|s_i(P) - s_i(Q)| \le H\alpha(1+\beta)d(P,Q)$.

Let $HS(P,Q,\gamma)$ denote the condition that $(|s_i(P) - s_i(Q)| \le \gamma \wedge |c_i(P) - c_i(Q)| \le \gamma)$ holds for all i. From Theorem 1, the following property holds: if $HS(P,Q,\gamma)$ does not hold for $\gamma = H\alpha(1+\beta)\delta$, then $d(P,Q) > \delta$. Thus, if $HS(P,Q,\gamma)$ does not hold, it can be concluded that $d(P,Q) > \delta$ without computing *rmsd* $(d(P,Q))$. However, note that $d(P,Q) \le \delta$ does not necessarily hold if $HS(P,Q,\gamma)$ holds.

Note that $\boldsymbol{hs}(P)$ can be computed in $O(H)$ time, and condition $HS(P,Q,\gamma)$ can be tested in constant time since we assume that D is a fixed small integer.

3.3 Variants

In this subsection, we describe several variants of HASH(A). First, we describe HASH(B) and HASH(B'), which are obtained by replacing the centroid $\boldsymbol{c}$ of HASH(A) with other positions.

HASH(B):
$\boldsymbol{hs}(P) = (c'_1(P), s'_1(P), \cdots, c'_D(P), s'_D(P))$, where

$$c'_i(P) = \alpha \sum_{k=1}^{H} \|\boldsymbol{p}^k - \boldsymbol{d}\| (\cos(\frac{2\pi i(k-1)}{H}) + \beta),$$

$$s'_i(P) = \alpha \sum_{k=1}^{H} \|\boldsymbol{p}^k - \boldsymbol{d}\| (\sin(\frac{2\pi i(k-1)}{H}) + \beta),$$

and $\boldsymbol{d} = \sum_{k=1}^{L} \boldsymbol{p}^k$ (i.e., $\boldsymbol{d}$ is the centroid of $\{\boldsymbol{p}^1, \cdots, \boldsymbol{p}^L\}$).

HASH(B'):
$\boldsymbol{hs}(P) = (c''_1(P), s''_1(P), \cdots, c''_D(P), s''_D(P))$, where

$$c''_i(P) = \alpha \sum_{k=1}^{H} \|\boldsymbol{p}^k - \boldsymbol{e}\| (\cos(\frac{2\pi i(k-1)}{H}) + \beta),$$

$$s''_i(P) = \alpha \sum_{k=1}^{H} \|\boldsymbol{p}^k - \boldsymbol{e}\| (\sin(\frac{2\pi i(k-1)}{H}) + \beta),$$

and $\boldsymbol{e} = \sum_{k=N-L+1}^{N} \boldsymbol{p}^k$ (i.e., $\boldsymbol{e}$ is the centroid of $\{\boldsymbol{p}^{N-L+1}, \cdots, \boldsymbol{p}^N\}$).

Next, we describe HASH(A+B) and HASH(A+B+B'), each of which is a combination of the vectors described above.

HASH(A+B):
$\boldsymbol{hs}(P) = (c_1(P), s_1(P), \cdots, c_D(P), s_D(P),$
$c'_1(P), s'_1(P), \cdots, c'_D(P), s'_D(P)).$

HASH(A+B+B'):
$\boldsymbol{hs}(P) = (c_1(P), s_1(P), \cdots, c_D(P), s_D(P),$
$c'_1(P), s'_1(P), \cdots, c'_D(P), s'_D(P),$
$c''_1(P), s''_1(P), \cdots, c''_D(P), s''_D(P)).$

All hash vectors described above use the distances from a special point, while HASH(C) described below uses the distances between points in the input fragment.

Proceedings of the 28th Hawaii International Conference on System Sciences (HICSS '95)
1060-3425/95 $10.00 © 1995 IEEE

HASH(C):
$\boldsymbol{hs}(P) = (\hat{c}_1(P), \hat{s}_1(P), \cdots, \hat{c}_D(P), \hat{s}_D(P))$, where

$$\hat{c}_i(P) = \alpha \sum_{k=1}^{\frac{H}{2}} \|\boldsymbol{p}^i - \boldsymbol{p}^{i+\frac{H}{2}}\| (\cos(\frac{2\pi i(k-1)}{H}) + \beta),$$
$$\hat{s}_i(P) = \alpha \sum_{k=1}^{\frac{H}{2}} \|\boldsymbol{p}^i - \boldsymbol{p}^{i+\frac{H}{2}}\| (\sin(\frac{2\pi i(k-1)}{H}) + \beta).$$

HASH(A+C):
$\boldsymbol{hs}(P) = (c_1(P), s_1(P), \cdots, c_D(P), s_D(P), \hat{c}_1(P), \hat{s}_1(P), \cdots, \hat{c}_D(P), \hat{s}_D(P))$.

Finally, we describe HASH(X). While all vectors described above correspond to a 1D Fourier spectrum, HASH(X) corresponds to a 2D Fourier spectrum. HASH(X) is similar to (a low frequency part of) a 2D Fourier spectrum of the distance map.

HASH(X):
$\boldsymbol{hs}(P) = (cc_{11}(P), cs_{11}(P), sc_{11}(P), ss_{11}(P), cc_{12}(P), cs_{12}(P), sc_{12}(P), ss_{12}(P), \cdots, cc_{DD}(P), cs_{DD}(P), sc_{DD}(P), ss_{DD}(P))$, where

$$cc_{ij} = \alpha \sum_{k=1}^{H} \sum_{h=1}^{H} L_{hk}(\cos(\frac{2\pi i(k-1)}{H}) \cos(\frac{2\pi j(h-1)}{H}) + \beta),$$
$$cs_{ij} = \alpha \sum_{k=1}^{H} \sum_{h=1}^{H} L_{hk}(\cos(\frac{2\pi i(k-1)}{H}) \sin(\frac{2\pi j(h-1)}{H}) + \beta),$$
$$sc_{ij} = \alpha \sum_{k=1}^{H} \sum_{h=1}^{H} L_{hk}(\sin(\frac{2\pi i(k-1)}{H}) \cos(\frac{2\pi j(h-1)}{H}) + \beta),$$
$$ss_{ij} = \alpha \sum_{k=1}^{H} \sum_{h=1}^{H} L_{hk}(\sin(\frac{2\pi i(k-1)}{H}) \sin(\frac{2\pi j(h-1)}{H}) + \beta),$$
$$L_{hk} = \|\boldsymbol{p}^k - \boldsymbol{p}^h\|.$$

Here, we consider theoretical properties of the variants. The following lemma can be proved in a similar way as in Lemma 1.

Lemma 2: Let $P = (\boldsymbol{p}^1, \cdots, \boldsymbol{p}^H)$, $Q = (\boldsymbol{q}^1, \cdots, \boldsymbol{q}^H)$, $\boldsymbol{d}^1 = \frac{1}{L}\sum_{i=1}^{L} \boldsymbol{p}^i$ and $\boldsymbol{d}^2 = \frac{1}{L}\sum_{i=1}^{L} \boldsymbol{q}^i$. Then,

$$|\sum_{i=1}^{H} \|\boldsymbol{p}^i - \boldsymbol{d}^1\| - \sum_{i=1}^{H} \|\boldsymbol{q}^i - \boldsymbol{d}^2\|| \leq (1 + \frac{H}{L}) H\, d(P, Q).$$

From Lemma 2, the following theorem is immediately proved, which shows that HASH(B) satisfies conditions (A) and (B).

Theorem 2: For all i, $|c'_i(P) - c'_i(Q)| \leq (1 + \frac{H}{L})H\alpha(1+\beta)d(P,Q)$ and $|s'_i(P) - s'_i(Q)| \leq (1 + \frac{H}{L})H\alpha(1+\beta)d(P,Q)$.

It is not difficult to see that similar properties hold for the other vectors except HASH(X). However, we have not yet proved a similar property for HASH(X).

3.4 Substructure search using hash vectors

Using $\boldsymbol{hs}(P)$, the substructure search can be done quickly. For a quick substructure search, two phases are required: the *preprocessing phase* and the *search phase*.

First, we describe the preprocessing phase. It is done whenever a new protein structure is registered in a database. Let Q^j be a new protein structure being registered, where we assume that Q^j consists of n points. Then, hash vectors are computed in the following way.

(1) For $i = 1$ to $n - H + 1$, repeat (2).
(2) Compute $\boldsymbol{hs}(Q^j_{i,i+H-1})$.

Note that H is a constant and $H = 40$ is used in the current version.

The computed hash vectors are stored in a database along with the position data of Cα atoms (see Fig. 3). Although it takes $O(Hn)$ time to compute the hash vectors for each Q^j, the time can be ignored since the hash vectors for Q^j must be computed only when Q^j is stored in a database. The memory space required to store the hash vectors for each Q^j is $O(n)$, because the size of a hash vector can be considered as a constant.

Next, we describe the search phase. In the search phase, a pattern structure P is input, where we assume that the length of P is $m \geq H$. Then, a substructure search is done in the following way. First, $\boldsymbol{hs}(P_{1,H})$ is computed. Next, for all fragments $Q^j_{i,i+H-1}$, $\boldsymbol{hs}(P_{1,H})$ and $\boldsymbol{hs}(Q^j_{i,i+H-1})$ are compared. If condition $HS(P_{1,H}, Q^j_{i,i+H-1}, \gamma)$ is satisfied, $d(P, Q^j_{i,i+m-1})$ is computed and tested. Otherwise, $d(P, Q^j_{i,i+m-1})$ is not computed, but the next fragment is tested. The following summarizes the procedure for the search phase (see also Fig. 4), where $|Q^j|$ denotes the number of Cα atoms in a protein structure Q^j, and we assume that there are N structures in a database.

(1) Compute $\boldsymbol{hs}(P_{1,H})$.
(2) For $j = 1$ to N, repeat (3)-(5).
(3) For $i = 1$ to $|Q^j| - m + 1$, repeat (4)(5).
(4) If condition $HS(P_{1,H}, Q^j_{i,i+H-1}, \gamma)$ holds, then do (5).
(5) If $d(P, Q^j_{i,i+m-1}) \leq \delta$, then output Q^j and try next j.

Proceedings of the 28th Hawaii International Conference on System Sciences (HICSS '95)
1060-3425/95 $10.00 © 1995 IEEE

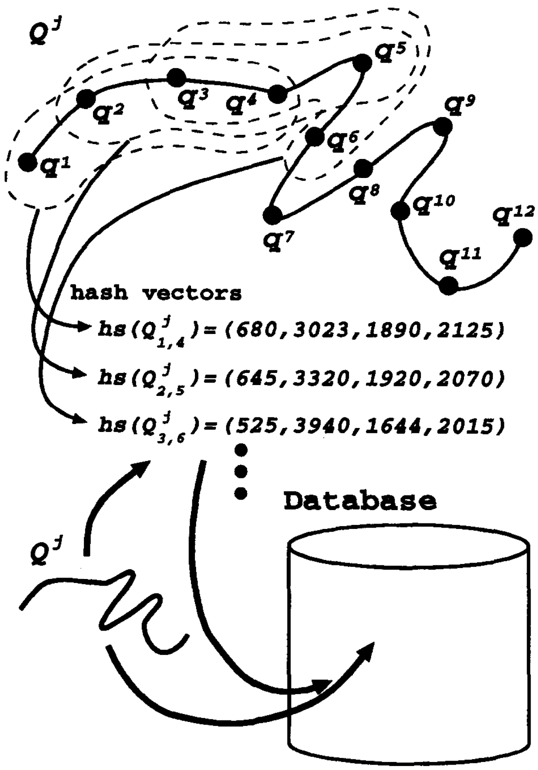

Fig. 3. The preprocessing phase ($H = 4$).

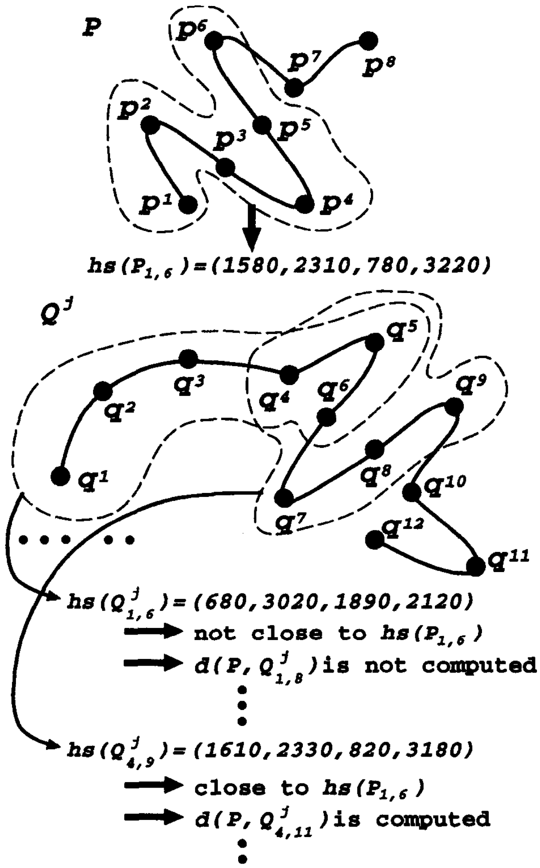

Fig. 4. The search phase ($H = 6$).

Here, we consider the computation time for the above procedure. We consider the time for (2)-(5), because the time for (1) is much smaller. It is expected that (5) is not executed for most i. Thus, the time for each protein structure Q^j is expected to be $O(n)$ in most cases. Thus, the search time for N protein structures in a database is expected to be $O(Nn)$ in most cases.

Next, we consider the parameter γ. From Theorem 1, $\gamma = H\alpha(1+\beta)\delta$ should be used (in the case of $m = H$). However, experimental results show that it does not fail to find similar fragments even if a much smaller value is used. It is obvious that the search time is reduced if a smaller γ is used. Thus, the value for γ should be determined based on experimental results.

The search time might be reduced if hash vectors are stored in a special data structure in which similar vectors can be retrieved quickly [18]. However, we did not implement such a data structure since it is too complicated and does not seem to be practical.

4 Experimental results

Experiments have been done using PDB data [3]. Although PDB data contains various kinds of information, only positions of Cα atoms are used. All algorithms are implemented in the C language on a SUN SPARC STATION-10 (a UNIX workstation).

The new hashing methods are compared with the previous and the naive ones in Table 1. NV denotes the naive method described in Subsection 2.2. LS denotes the least-squares hashing method [1]. Both A and A' denote HASH(A), where $D = 4$ is used in A and $D = 8$ is used in A'. B, A+B and A+B+B' denote HASH(B), HASH(A+B) and HASH(A+B+B'), respectively, where $D = 4$ and $L = \frac{H}{4}$ is used in each case. A+C denotes HASH(A+C), where $D = 4$ is

Table 1. Comparison of the new hashing methods.

DATA	NUM	NV	LS	A	A'	B	A+B	A+B +B'	A+C	X
4HHB(A) (50-94)	57	63.0	12.1 12.6 %	4.9 6.7 %	5.2 6.6 %	4.0 5.0 %	2.5 3.0 %	1.5 1.3 %	3.0 3.5 %	4.2 5.2 %
7LZM (35-80)	86	64.0	23.7 25.5 %	8.8 12.7 %	9.4 12.5 %	5.1 6.5 %	2.3 2.5 %	1.3 0.9 %	3.1 3.6 %	4.7 5.9 %
1R69 (5-55)	6	67.5	24.8 25.5 %	6.7 9.1 %	7.2 9.0 %	8.6 11.2 %	3.8 4.5 %	1.5 1.1 %	4.5 5.4 %	4.7 5.8 %
3ADK (115-170)	5	70.9	5.1 4.4 %	6.8 8.7 %	7.2 8.7 %	4.9 5.7 %	3.4 3.8 %	1.8 1.6 %	3.5 3.9 %	4.7 5.5 %
8LDH (150-194)	10	63.2	8.1 8.0 %	1.2 1.2 %	1.2 1.1 %	1.1 0.7 %	0.6 0.3 %	0.5 0.1 %	0.7 0.3 %	1.5 1.5 %

used. X denotes HASH(X), where $D = 3$ is used.

Each item in DATA denotes a filename (denoting a protein structure) of PDB data, where chain A is used in the case of 4HHB. Each pair of numbers in parentheses denotes the range of positions of a fragment P. Each item in NUM denotes the number of protein structures Q^j's each of which contains a fragment $Q^j_{i,i+m-1}$ such that $d(P, Q^j_{i,i+m-1}) \leq \delta$. For each algorithm and each pattern fragment, search time (CPU time (sec)) among all structures in a database is shown, where 811 structures are used and all structural data are stored in main memory of the workstation. A percentage of indices, for which *rmsd* is computed, is described too. In each algorithm except HASH(X), preprocessing (i.e., computation of hash vectors) for all structures was completed in a few minutes, so that it can be neglected. In HASH(X), it took more than an hour. However, it may be allowed since preprocessing must be done only once.

The following parameters were used: $H = 40$, $\alpha = 20.0$, $\beta = 0.5$, $\delta = 4.0(\AA)$ and $\gamma = 1200.0$, where $\alpha = 1.0$ was used in HASH(X). In general, it is expected that search time is reduced if a larger value of D is used. However, comparison of A and A' shows that search time increases although a larger D is used, because the time for comparing hash vectors increases, while the percentage of indices, for which *rmsd* is computed, is reduced at most 0.2%. Thus, $D = 4$ is used. Although $\gamma \ll H\alpha(1+\beta)\delta$ was used, each method except LS could find all structures each of which contained a fragment $Q^j_{i,i+m-1}$ such that $d(P, Q^j_{i,i+m-1}) \leq \delta$. LS failed to find 3 structures in the case of 3ADK. Moreover, LS took longer than the other hashing methods in most cases. Thus, it is proved that the new hashing methods are much more useful than the least-squares hashing method.

From Table 1, it is seen that the following relation holds approximately:

$$\text{A+B+B'} \succ \text{A+B} \approx \text{A+C} \succ \text{X} \succ$$
$$\text{A} \approx \text{A'} \approx \text{B} \succ \text{LS} \succ \text{NV},$$

where $x \succ y$ denotes that x is faster than y, and $x \approx y$ denotes that x is as fast as y. Thus, it can be said that we had better combine different types of hash vectors. In each of the new hashing methods, it can be seen that the search time was reduced considerably compared with the naive method. Especially, it is seen that HASH(A+B+B') is 30 $\sim$ 100 times faster than the naive method. Thus, it is proved that the new hashing methods, especially HASH(A+B+B'), are very effective.

5 PROTEIX

One of the hashing methods (HASH(A+B+B')) was already included in PROTEIX (PROTEIn database management system on uniX), which is a database system for 3D protein structures. PROTEIX is being developed to supply a tool for molecular biologists who analyze 3D structural patterns of proteins. Here, we overview PROTEIX.

5.1 Overview of the system

Fig. 5 illustrates an overview of the PROTEIX system. PROTEIX consists of three parts:

(A) An in-memory database consisting of protein structures,

(B) An interactive graphic interface for displaying 3D protein structures and inputing commands,

Proceedings of the 28th Hawaii International Conference on System Sciences (HICSS '95)
1060-3425/95 $10.00

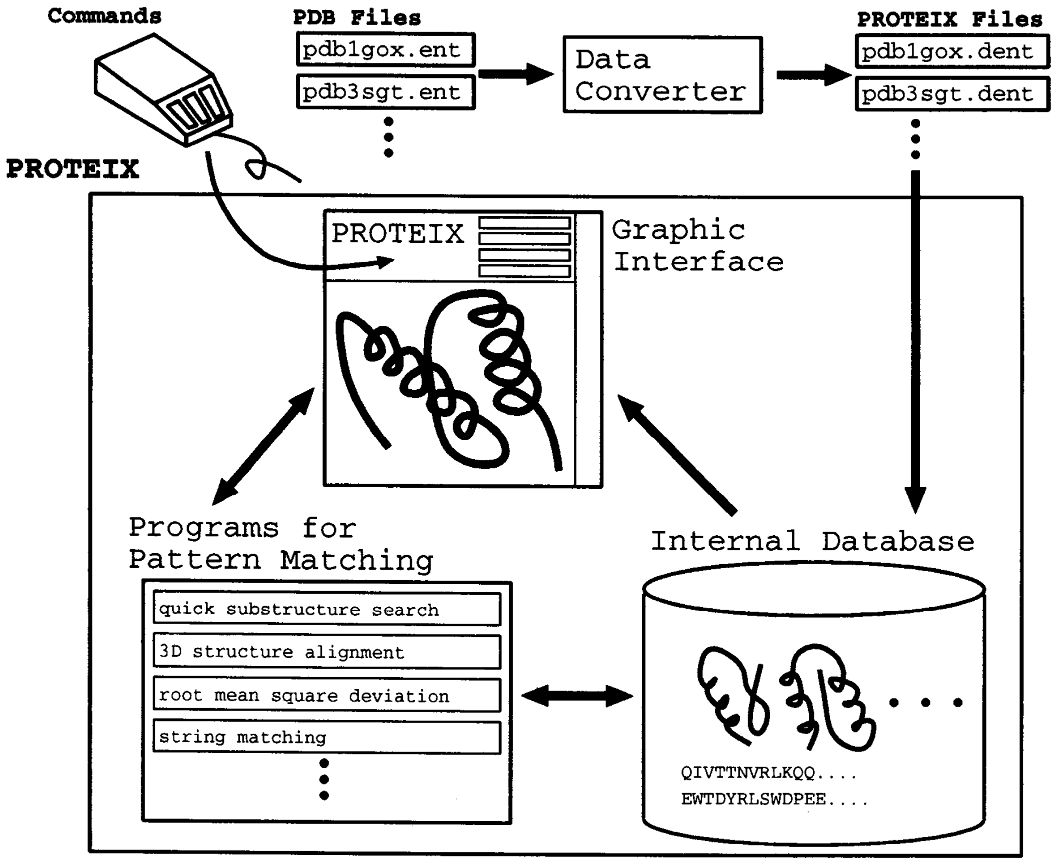

Fig. 5. Overview of PROTEIX.

(C) A set of pattern matching programs for 3D protein structures and amino acid sequences.

All parts of PROTEIX are written in the C language, while X-window (X11R5) and Motif have been adopted for the graphic interface. While PROTEIX is now working on SUN SPARC workstations, it will work on various UNIX workstations with slight modifications, because it has high portability.

5.2 Pattern matching facilities

The most important feature of PROTEIX is that it has powerful pattern matching programs for 3D protein structures. It has three pattern matching programs for 3D protein structures: a conventional *rmsd* fitting program, a substructure search program using the hashing method described in this paper, and an alignment program for two 3D structures. For the alignment program, an algorithm based on the dynamic programming technique is used, which we have already proposed [1]. To allow quick substructure searching, all protein structure data are stored in main memory. Since compact data structures are used, more than 1000 structures can be stored within 32Mbyte main memory.

The following is a typical method of using the above facilities. Assume that we want to study protein structures similar to some protein structure S. We first pick some small fragment P from S, and find structures Q^j's each of which contains a fragment similar to P, using the substructure search program. Then, for each structure Q^j, we study structural similarities between S and Q^j, using the structural alignment program and the *rmsd* fitting program.

Proceedings of the 28th Hawaii International Conference on System Sciences (HICSS '95)
1060-3425/95 $10.00 © 1995 IEEE

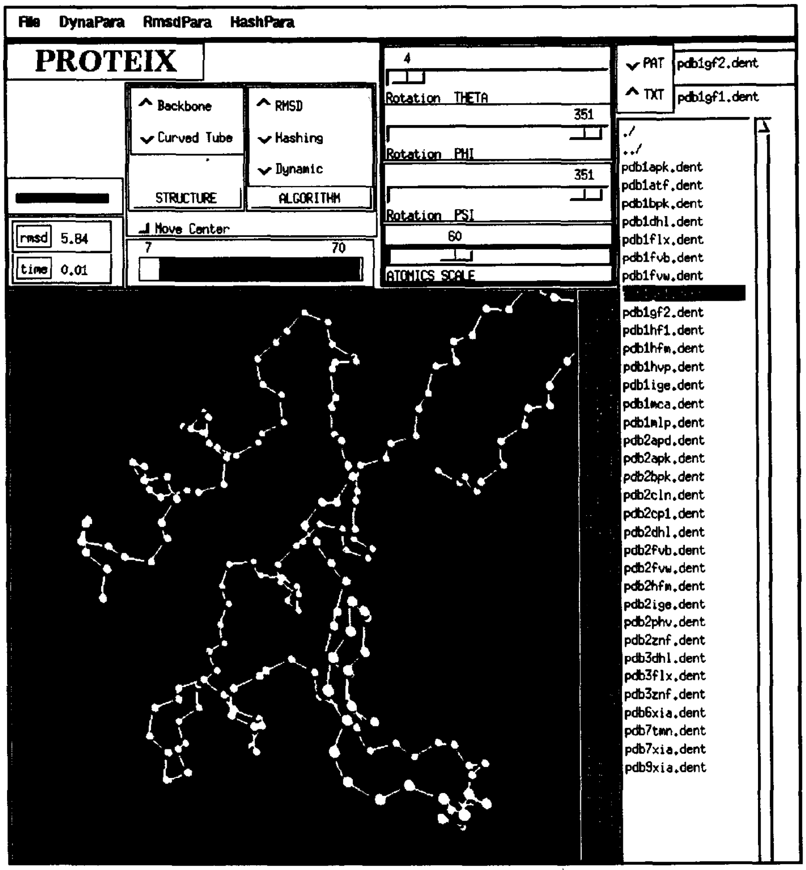

Fig. 6. Snapshot of PROTEIX (display mode M1).

5.3 Graphic interface

PROTEIX has an interactive color graphic interface for displaying 3D protein structures and inputing commands (see Fig. 6). It has two modes (M1 and M2) for displaying a 3D structure. In mode M1, Cα atoms (or the carbon and nitrogen atoms in a backbone chain) and bonds connecting them are displayed (Fig. 6). In mode M2, an outline of the shape of a backbone chain is displayed as a curved tube. Note that proteins are displayed on a color display although only gray colors are used in Fig. 6. Moreover, after applying a pattern matching program to two structures, two structures are shown superimposed. Of course, in each mode, structures can be rotated and scaled down or up.

PROTEIX allows the user to use fragments of structures, by specifying the start and end positions of residues. It is useful to look at details of the structure as well as to specify a fragment for a substructure search.

The graphic interface of PROTEIX is designed so that users not familiar with UNIX can use it. Currently, all commands are input using the 'mouse' device only. Such commands as rotation and scaling can be done by clicking the mouse button at an appropriate position of the bar. The user can specify PDB files using the file selection box, which appears on the right of the display window. A PDB file can be specified by clicking the mouse at the position where the file name appears. Because there are a lot of files in PDB, the hierarchical file structure (directory structure) of UNIX is adopted. Files found by a substructure search are also displayed in the file selection box.

Proceedings of the 28th Hawaii International Conference on System Sciences (HICSS '95)
~~1060-3425/95 $10.00 © 1995 IEEE~~

6 Conclusion

In this paper, we have described new hashing methods for quick substructure searching as well as the PROTEIX database management system for 3D protein structures. The proposed hashing methods have desirable properties, which are proved theoretically. Moreover, experimental results show that the methods are very fast and effective. It seems that making a considerable improvement on the methods is very difficult. Thus, such an improvement is a challenging open problem.

One of the proposed methods was already included in the PROTEIX system. We believe that PROTEIX is a useful tool for molecular biologists who study structural relations among proteins. For that purpose, we will continue to improve PROTEIX. The followings are future plans for PROTEIX:

- A graphic editor which allows user to input new structures or modify existing structures,
- Structural alignment algorithms/programs for more than two structures,
- Interfaces to other systems/programs for protein structures.

Acknowledgement

This research was partially supported by the Grant-in-Aid for Scientific Research on Priority Areas, "Genome Informatics", of the Ministry of Education, Science and Culture of Japan.

References

[1] T. Akutsu, "Efficient and robust three-dimensional pattern matching algorithms using hashing and dynamic programming techniques", *Proc. 27th Hawaii International Conference on System Sciences*, 1994, pp. 225-234.

[2] G. J. Barton and M. J. E. Sternberg, "LOPAL and SCAMP: techniques for the comparison and display of protein structures", *J. Molecular Graphics* **6** (1988) 190-196.

[3] F. C. Bernstein, T. F. Koetzle, G. J. B. Williams, E. F. Meyer Jr., M. D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi and M. Tasumi, "The Protein Data Bank: A computer-based archival file for macromolecular structures", *J. Molecular Biology* **112** (1977) 535-542.

[4] C. Branden and J. Tooze, *Introduction to Protein Structure* (Garland Publishing, 1991).

[5] W. Kabsch, "A solution for the best rotation to relate two sets of vectors", *Acta. Cryst.* **A32** (1976) 922-923.

[6] A. Kalvin, E. Schonberg, J. T. Schwartz and M. Sharir, "Two-dimensional, model-based, boundary matching using footprints", *Int. J. Robotics Research* **5** (1986) 28-55.

[7] E. Kishon and H. Wolfson, "3-D curve matching", *Proc. AAAI Workshop on Spatial Reasoning and Multi-Sensor Fusion*, 1987, pp. 250-261.

[8] R. Nussinov and H. J. Wolfson, "Efficient detection of three-dimensional structural motifs in biological macromolecules by computer vision techniques", *Proc. Natl. Acad. Sci. (USA)* **88** (1991) 10495-10499.

[9] C. A. Orengo and W. R. Taylor, "A rapid method of protein structure alignment", *J. Theoretical Biology* **147** (1990) 517-551.

[10] S. T. Rao and M. G. Rossmann, "Comparison of super-secondary structures in proteins", *J. Molecular Biology* **76** (1973) 241-256.

[11] M. G. Rossmann and P. Argos, "Exploring structural homology of proteins", *J. Molecular Biology* **105** (1976) 75-95.

[12] R. B. Russell and G. J. Barton, "Multiple protein sequence alignment from tertiary structure comparison: assignment of global and residue confidence levels", *PROTEINS: Structure, Function, and Genetics* **14** (1992) 309-323.

[13] J. T. Schwartz and M. Sharir, "Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves", *Int. J. Robotics Research* **6** (1987) 29-44.

[14] A. Tamura and M. Hirota, "A study on automatic methods for finding relationship between structural patterns and sequence patterns of proteins", *Bachelor Thesis* (in Japanese), (Science University of Tokyo, 1994).

[15] W. R. Taylor and C. A. Orengo, "Protein structure alignment", *J. Molecular Biology* **208** (1989) 1-22.

[16] J. M. Thornton and S. P. Gardner, "Protein motifs and data-base searching", *Trends in Biochemical Science* **14** (1989) 300-304.

[17] G. Vriend and C. Sander, "Detection of common three-dimensional substructures in proteins", *PROTEINS: Structure, Function, and Genetics* **11** (1991) 52-58.

[18] D. E. Willard, "New data structures for orthogonal range queries", *SIAM J. Computing* **14** (1985) 232-253.