Comparison Algorithm of Protein Sequences by Introducing a Lookup Table

Shin-ichi NAKAYAMA* and Masayuki YOSHIDA

University of Library and Information Science
1-2, Kasuga, Tsukuba-shi, Ibaraki-ken, 305, Japan
nakayama@ulis.ac.jp, myoshida@ulis.ac.jp

(Received : August 5, 1996 ; Accepted for publication : October 7, 1996)

The comparison algorithm of protein sequences by Murata was modified by considering only similarities between the same amino acid types and by introducing a lookup table. The modified algorithm calculates the similarity score between two protein sequences and aligns them. Computational times for calculating similarity scores against some proteins to all proteins in the Protein Sequence Database of the PIR by the system implementing our algorithm are about ten times shorter than those achieved by Murata. The alignments by our algorithm with a gap penalty of 18 are similar to those by Murata. The computational times by our system are two to three times slower, but the similarity scores are more sensitive than those by FASTA.

Keywords: alignment, similarity score, amino acid sequence, maximum array, lookup table

Information obtained by comparing protein sequences is thought to be useful in studies of molecular biology. Thus, various algorithms comparing two protein sequences have been developed. Needleman and Wunsch [1] presented a dynamic programming algorithm, in which a comparison matrix of two protein sequences, consisting of similarity weights between two amino acids in the sequences, was transformed into a maximum match matrix. From this matrix, a similarity score and an alignment between the two protein sequences were obtained. Although this algorithm gives reasonable similarity scores and alignments, implementation of the algorithm requires computational times proportional to \((mn^2 + m'n)/2\), where \(m\) and \(n\) are the respective lengths of the two protein sequences. Thus, the algorithm is unsuitable for searching similar proteins in a large scale database and for obtaining multiple alignments which necessitate many
aligning operations.

Murata [2] modified the algorithm of Needleman and Wunsch by introducing a maximum array for a column of interests in the original comparison matrix. Elements of the array consist of maximum values in submatrixes, which are the lower right-hand areas viewed from each point in the column. His algorithm reduces the computational times to values proportional to \(nm\) and gives the same scores and alignments obtained by Needleman and Wunsch.

If similarities between different types of amino acids were not taken into account, computation involving such similarities could be eliminated from the above algorithm. Thus, we modified the algorithm of Murata by introducing a lookup table and achieved rapid computational times. In this paper we describe our algorithm and discuss the effect of gap penalty scores on alignments of two protein sequences and the sensitivity of our similarity scores.

**System and methods**

**Algorithm:** The \(i\)-th residue in a sequence \(a\) and the \(j\)-th residue in a sequence \(b\) are denoted as \(Aa(i)\) and \(Ab(j)\), respectively. The original comparison matrix \(M\) is composed of elements \(M(i, j)\), to which weights of similarity between amino acids \(Aa(i)\) and \(Ab(j)\) are initially assigned. There are many pathways to descend elements of the comparison matrix from the left upper corner to the right bottom one. A step from \(M(i, j)\) to \(M(i+1, j+1+x)\) indicates a deletion of \(x\) pieces of amino acid residues in the sequence \(a\) or indicates an insertion of \(x\) pieces of amino acid residues at \(j+1\) to \(j+1+x\) in the sequence \(b\). In each pathway, the sum of the values of elements \(M(i, j)\) and the minus values of gap penalties, which are imposed for steps including deletion or insertion of amino acid residues, is calculated. The maximum value among the pathways sums is assigned as the similarity score between the two sequences. An alignment of two sequences is obtained as a series of amino acid pairs derived from the pathway having the maximum value.

To obtain the similarity scores Needleman and Wunsch developed the following algorithm[1]:

```c
for ( i = m ; i ≥ 1 ; i-- ) {
    for ( j = n ; j ≥ 1 ; j-- ) {
        max_d = M(i+1, j+1);
        for ( x=m; x ≥ i+2 ) {
            max_d = max[max_d, M(x, j+1) -γ ];
        }
        for ( y=n; y ≥ j+2 ) {
            max_d = max[max_d, M(i+1, y) -γ ];
        }
    }
}
```
where \( m \) and \( n \) are lengths of the sequence \( a \) and \( b \), respectively, and \( \gamma \) is a gap penalty. The maximum value of the elements of the transformed matrix \( M \) is the similarity score between two sequences and the alignment is obtained by matching amino acids from the place of the maximum value element to that of the bottom right corner along the ridges of the transformed matrix.

To the above algorithm, Murata introduced \[2\] a maximum array \( \mu \) composed of \( n+1 \) elements, in which the element \( \mu(j+1) \) was expressed by the maximum value of \( M(x, y) \) in the area of \( i \leq x \leq m \) and \( j \leq y \leq n \). Needleman and Wunsch's algorithm was improved by using the maximum array as follows:

\begin{verbatim}
for(i=m;i>=1;i--)
  for(j=n;j>=1;j--)
    M(i,j) = M(i,j) + max[M(i+1,j+1), \mu(j+1) - \gamma];
    if(j<n) {
      \mu(j+1) = max\{M(i,j+1), \mu(j+1), \mu(j+2)\};
    }
\end{verbatim}

Implementation of this algorithm to obtain the similarity score requires computational time proportional to \( mn \).

Comparison to obtain the elements of the maximum array at the 0-element in the original comparison matrix is unnecessary. As the number of amino acids types in protein sequences is 20, the probability of existence of any particular amino acid type in a protein sequence is \( 1/20 \). Thus, if the weights of the similarity between different types of amino acids are set to 0, the proportion of the elements with 0 to all elements of the original comparison matrix is \( 19/20 \). As we could transform the original comparison matrix by skipping those elements, the computational time for obtaining a similarity score between two sequences decreases to \( 1/20 \) of Murata’s algorithm. Lookup tables are useful to carry out the above idea \[3\]. The lookup table of a sequence is constructed for each type of amino acid using its position in the sequence. For example, a sequence \( z \), consisting of NLWRNWN, gives the table of \( T_z(L,1)=2, T_z(N,1)=1, T_z(N,2)=5, T_z(N,3)=7, T_z(R,1)=4, T_z(W,1)=3, \) and \( T_z(W,2)=6 \). When an amino acid, for example N, is given, the positions of non-zero elements, 1, 5, and 7, are obtained from this lookup table.

Using this lookup table, we modified Murata's algorithm as follows:
for (i=m; i≥1; i--) {
    new_m = 0;
    count = 1; j = Tb(Aa(i), count);
    while (j ≠ NULL) {
        δ(i-j) = W(Aa(i)) + max[δ(i-j), μ(j+1) - γ];
        μ(j+1) = max[μ(j+1), new_m, δ(i-j)];
        before_j = j; new_m = μ(j+1);
        count++; j = Tb(Aa(i), count);
        for (k = before_j; k > j + 1 && μ(before_j+1) > μ(k) ; k--) {
            μ(j) = μ(before_j+1);
        }
    }
    for (k = before_j; k > 0 && μ(before_j+1) > μ(k) ; k--) {
        μ(j) = μ(before_j+1);
    }
}

where W(x) is a similarity score between the same amino acid x, and δ is the array, which consists of the maximum values in element of each diagonal of a matrix during transformation. The array of μ is referred to distinguish the occurrence of the insertion and deletion of amino acids and the array of δ is referred to distinguish the occurrence of the exchange of amino acids. In this algorithm the lookup table was made from the last position in the protein sequence, and NULL, indicating end of data, is added to the table as the final element. For example, sequence z described above gives the table of Tz(A,1)=NULL, ... Tz(L,1)=2, Tz(L,2)=NULL, ... Tz(N,1)=7, Tz(N,2)=5, Tz(N,3)=1, Tz(N,4)=NULL, ... Tz(R,1)=4, Tz(R,2)=NULL, ... Tz(W,1)=6, Tz(W,2)=3, Tz(W,3)=NULL, ... Tz(Y,1)=NULL.

Using the above modified algorithm, the alignment of two sequences is obtained. In the first step the next positions of pathway in each element are stored in Pi and Pj as follows:

for (i = m; i ≥ 1; i--) {
    new_m = 0;
    count = 1; j = Tb(Aa(i), count);
    while (j ≠ NULL) {
        δ(i-j) = W(Aa(i)) + max[δ(i-j), μ(j+1) - γ];
        if (δ(i-j) > μ(j+1) - γ){
            Pi(i, j) = dpi(i-j); Pj(i, j) = dpj(i-j);
        }else{
            Pi(i, j) = mpi(j+1); Pj(i, j) = mpj(j+1);
        }
    }
}


\[ dpi(i-j) = i; \ dpj(i-j) = j; \]
\[ \mu(j+1) = \max[\mu(j+1), new_m, \delta(i-j)]; \]
\[ \text{if } (\text{new}_m > \mu(j+1) \ \&\& \text{new}_m > \delta(i-j)) \{ \]
\[ \ mpi(j+1) = \text{new}_p; \ mpj(j+1) = \text{new}_p; \]
\[ \} \text{ else } \{ \]
\[ \ \text{if } (\delta(i-j) > \mu(j+1) ) \{ \]
\[ \ mpi(j+1) = P_i(i, j); \ mpj(j+1) = P_j(i, j); \]
\[ \} \]
\[ \text{before}_j = j; \ \text{new}_m = \mu(j+1); \]
\[ \text{new}_p = mpi(j+1); \ \text{new}_p = mpj(j+1); \]
\[ \text{count}++; \ j = T_b(A_a(i), \text{count}); \]
\[ \text{for } (k = \text{before}_j; k > j + 1 \ &\& \ \mu(\text{before}_j+1) > \mu(k); \ k-- ) \{ \]
\[ \ \mu(j) = \mu(\text{before}_j+1); \]
\[ \ mpi(k) = mpi(\text{before}_j+1); \ \text{mp}(k) = mpj(\text{before}_j+1); \]
\[ \} \]
\[ \text{for } (k = \text{before}_j; k > 0 \ &\& \ \mu(\text{before}_j+1) > \mu(k); \ k-- ) \{ \]
\[ \ \mu(j) = \mu(\text{before}_j+1); \]
\[ \ mpi(k) = mpi(\text{before}_j+1); \ \text{mp}(k) = mpj(\text{before}_j+1); \]
\[ \} \]

where \( P_i \) and \( P_j \) indicate the next positions of low and those of column, respectively. In the second step two sequences are aligned as follows:

\[ ip = mpi(1); \ \text{jp} = mpj(1); \]
\[ \text{nsq} = i = j = 1; \]
\[ \text{while } (ip - \text{jp} > \text{nsq} ) \{ \]
\[ sq_{A_a}(\text{nsq}) = A_a(i); \ \text{sq}_{A_b}(\text{nsq}) = '\text{-}'; \ \text{nsq}++; \ i++; \]
\[ \} \]
\[ \text{while } (lp - ip > \text{nsq} ) \{ \]
\[ sq_{A_a}(\text{nsq}) = '\text{-}'; \ \text{sq}_{A_b}(\text{nsq}) = A_b(j); \ \text{nsq}++; \ j++; \]
\[ \} \]
\[ \text{while } (ip \geq i \ &\& \ \text{jp} \geq j) \{ \]
\[ sq_{A_a}(\text{nsq}) = A_a(i); \ sq_{A_b}(\text{nsq}) = A_b(j); \ \text{nsq}++; \ i++; \ j++; \]
\[ \} \]
\[ ip = P_i(i-1, j-1); \ \text{jp} = P_j(i-1, j-1); \]
\[ \text{while } (m > i \ &\& \ n > j ) \{ \]
\[ \text{diff} = (ip - i) - (jp - j); \]
old_i = i; old_j = j;
if (diff > 0) {
    while ( old_i+diff > i ) {
        sq_Aa(nsq) = Aa(i); sq_Ab(nsq) = '-'; nsq++; i++;
    }
} else {
    while ( old_j+diff > j ) {
        sq_Aa(nsq) = '-'; sq_Ab(nsq) = Ab(j); nsq++; j++;
    }
}
while (ip ≥ i && jp ≥ j){
    sq_Aa(nsq) = Aa(i); sq_Ab(nsq) = Ab(j); nsq++; i++; j++;
}

where sq_Aa and sq_Ab are containers of alignment sequences obtained.

Experiments: To investigate the advantage of our algorithm in computation, systems for calculating the similarity scores between two protein sequences based on Murata's and our algorithms were constructed. Also, the systems for aligning two protein sequences based on both algorithms were made. In those systems, the similarity weights between two types of amino acids were taken from the PAM250 matrix [4]. Those programs were written in language C and run on the SONY NEWS-5000 workstation, which uses the R4000SC CPU (66.7MHz).

For the computation times, similarity scores between one of four sequences described below and all sequences in the Protein Sequence Database Release 44.00 of the Protein Identification Resource (PIR), which contains 12,404 proteins, were calculated and the times for those calculations were measured. The sequences examined are proteins of different lengths: cytochrome c (CCDG), trypsinogen (TRBOTR), helix-destabilizing protein (DDRT), and pepsinogen A (PEPS),
which have 104, 229, 320, and 388 amino acid residues, respectively.

For the sequence alignments, human hemoglobin α (HAHU) was aligned to human hemoglobin β (HBHU) and human myoglobin (MYHU), respectively, where the gap penalty was set to 6 in Murata's algorithm and set to 6, 12, and 18 in our one. Furthermore, CCDG was aligned to three distantly related cytochromes, where the gap penalty was set to 6 in Murata's algorithm and set to 18 in ours.

As to the similarity scores, those by Murata's and our algorithm, and the optimized score of FASTA [5] for the two sequences produced from CCDG with first 300 proteins in the protein sequence database were calculated. FASTA is used widely in searching protein sequence databases. In Murata's algorithm, similarities between different types of amino acids are taken into account. The gap penalty score is set to 6 in Murata's algorithm and is set to 18 in ours. In FASTA, the ktup is set to 1. The ktup parameter determines how many consecutive amino acids are required in match. The most sensitive searches can be done using ktup = 1.

**Results and Discussion**

The computation times for calculating similarity scores against one of four sequences previously described to all sequences in the PIR database are shown in Table 1.

The computational times by our algorithm are about ten times shorter than those obtained using the Murata algorithm. If the occurrence of each type of amino acid is even in each sequence used in this experiment, 19/20 of the original comparison matrix would be zero. Thus, the number of elements transformed to construct the maximum match matrix would decrease to 1/20 and the computational times with our algorithm would be 1/20 of those of Murata. The computation times with our algorithm obtained in this experiment are, however, only ten times shorter than those

| Proteins | Lengths | Murata[2] | This work |
|----------|---------|-----------|-----------|
| CCDG     | 104     | 353       | 39        |
| TRBOTR   | 229     | 765       | 73        |
| DDRT     | 331     | 1080      | 99        |
| PEPS     | 388     | 1290      | 128       |

*Table 1. Computational Times (s) for Calculating Similarity Scores by Both Algorithms*
achieved by Murata. This delay may be ascribed to unbalanced distributions in the numbers of each type of amino acids. A similar effect is observed in the FASTA [5]. There is little effect on the number of times that the values of the maximum array are renewed.

These results show that the similarity scores by our algorithm are computed faster than by Murata's algorithm. The similarity scores obtained by both algorithms give the same values, if the similarity weights between different types of amino acids are set to 0 in Murata's algorithm.

The alignments obtained by Murata's and our algorithms with various gap penalties are shown in Figure 1. In smaller gap penalty scores, the alignments by our algorithm are found to include many more gaps than that of Murata. The number of gaps is affected by the magnitude of the gap penalty scores. If the score is set to 18, the alignments by our algorithm are almost equal to those of Murata. The results suggest that our algorithm can give proper alignments with the gap penalty scores of around 18. To confirm this estimation, some alignment pairs, obtained by both algorithms, were compared. The results are shown in Figure 2. The alignment pairs are almost equal and the estimation is confirmed. In these experiments Murata's similarity scores between those aligned proteins are about 1000 and it is suggested that these pairs of proteins have distantly related sequences. Thus, it is concluded that our algorithm, setting gap penalty score to 18, can give proper alignments between lowly similar proteins.

The same similarity scores and alignments could be obtained by the algorithm of sparse dynamic programming [6]. The algorithm is a modification of Wilber and Lipman's fragment alignment algorithm [7]. In the sparse dynamic programming algorithm, the array, called ACTIVE, is used to keep the comparison places to determine the maximum score pathway. The number of comparisons for obtaining similarity scores and alignments in the algorithm is similar to ours. However the sparse dynamic programming algorithm has the restriction that the weight of the gap penalty is a linear function of the difference of fragments. In the case where a constant value was used for the gap penalty score, our algorithm was advantageous.

The correlation between Murata's and our similarity scores is shown in Figure 3. Our similarity scores quickly decrease from 1350 to 300 against Murata's similarity scores in between 1350 and 900, while our scores gradually decrease from 300 to 150 against Murata's between 900 and 300. If two similarity scores have the same sensitivity the graph is a straight line descending to the left. Since similarities between different types of amino acids are not taken into account in our algorithm, our similarity scores have lower sensitivity in distantly related proteins compared to those of Murata.

The correlation between Murata's similarity scores and the FASTA's optimized scores is
Murata's algorithm with gap penalty = 6

**HBHU**

VHILPEEKSATVALWGVK---NDEVGGEALGRLLVYYPWTQRFFESFGDLSTPDAM GwenKPVKAKGKGLVAFLSGDLAYLDNLKGTATLSELHDCDKLHVDPENFR
VLSPADTNVKAAGWVKGHAGEYGAELERMFLSFFTTYTFPHF---DLS---GSAQVKGKGVKAVADTLNAHVDDMPANALSDLHRAHKLVDPPVNF
LLGNVLCVLCHAFKFGFTTPPQAAYKQVVGANALAHKY
LSSCPLLVTLaHLPaETPFAHAVSLDKFLASVSTVLTSKR

**MYHU**

GSLGDGEWQLVNVWVKVEdIPGQGEVLILRFKGHPTELEKDFKPKLHSDKEMKASEDLKHKGATVLATCGILKGGHHEE1KPLAQSKHTKHKPVKYIF
VLSPADTNVKAAGWVGHAHEYGAELERMFLSFFTTYTFPHF---DLS---GSAQVKGKGVKAVADTLNAHVDDMPANALSDLHRAHKLVDPPVNF
ISEC1IQVLQSKHPDGDADAGMAKNAPEFLRDMASNRYKELFGFGQ
LSSCPLLVTLaHLPaETPFAHAVSLDKFLASVSTVLTSKR

Our algorithm with gap penalty = 6

**HBHU**

VHILPEEKSATVALWGVK---NDEVGGEALGRLLVYYPWTQRFFESFGDLSTPDAM GwenKPVKAKGKGLVAFLSGDLAYLDNLKGTATLSELHDCDKLHVDPENFR
VLSPADTNVKAAGWVKGHAGEYGAELERMFLSFFTTYTFPHF---DLS---GSAQVKGKGVKAVADTLNAHVDDMPANALSDLHRAHKLVDPPVNF
LLGNVLCVLCHAFKFGFTTPPQAAYKQVVGANALAHKY
LSSCPLLVTLaHLPaETPFAHAVSLDKFLASVSTVLTSKR

**MYHU**

GSLGDGEWQLVNVWVKVEdIPGQGEVLILRFKGHPTELEKDFKPKLHSDKEMKASEDLKHKGATVLATCGILKGGHHEE1KPLAQSKHTKHKPVKYIF
VLSPADTNVKAAGWVGHAHEYGAELERMFLSFFTTYTFPHF---DLS---GSAQVKGKGVKAVADTLNAHVDDMPANALSDLHRAHKLVDPPVNF
ISEC1IQVLQSKHPDGDADAGMAKNAPEFLRDMASNRYKELFGFGQ
LSSCPLLVTLaHLPaETPFAHAVSLDKFLASVSTVLTSKR

Our algorithm with gap penalty = 12

**HBHU**

VHILPEEKSATVALWGVK---NDEVGGEALGRLLVYYPWTQRFFESFGDLSTPDAM GwenKPVKAKGKGLVAFLSGDLAYLDNLKGTATLSELHDCDKLHVDPENFR
VLSPADTNVKAAGWVKGHAGEYGAELERMFLSFFTTYTFPHF---DLS---GSAQVKGKGVKAVADTLNAHVDDMPANALSDLHRAHKLVDPPVNF
LLGNVLCVLCHAFKFGFTTPPQAAYKQVVGANALAHKY
LSSCPLLVTLaHLPaETPFAHAVSLDKFLASVSTVLTSKR

**MYHU**

GSLGDGEWQLVNVWVKVEdIPGQGEVLILRFKGHPTELEKDFKPKLHSDKEMKASEDLKHKGATVLATCGILKGGHHEE1KPLAQSKHTKHKPVKYIF
VLSPADTNVKAAGWVGHAHEYGAELERMFLSFFTTYTFPHF---DLS---GSAQVKGKGVKAVADTLNAHVDDMPANALSDLHRAHKLVDPPVNF
ISEC1IQVLQSKHPDGDADAGMAKNAPEFLRDMASNRYKELFGFGQ
PVNVKLLSCLLLTLaHLPaETPFAHAVSLDKFLASVSTVLTSKR

Our algorithm with gap penalty = 18

**HBHU**

VHILPEEKSATVALWGVK---NDEVGGEALGRLLVYYPWTQRFFESFGDLSTPDAM GwenKPVKAKGKGLVAFLSGDLAYLDNLKGTATLSELHDCDKLHVDPENFR
VLSPADTNVKAAGWVKGHAGEYGAELERMFLSFFTTYTFPHF---DLS---GSAQVKGKGVKAVADTLNAHVDDMPANALSDLHRAHKLVDPPVNF
LLGNVLCVLCHAFKFGFTTPPQAAYKQVVGANALAHKY
LSSCPLLVTLaHLPaETPFAHAVSLDKFLASVSTVLTSKR

**MYHU**

GSLGDGEWQLVNVWVKVEdIPGQGEVLILRFKGHPTELEKDFKPKLHSDKEMKASEDLKHKGATVLATCGILKGGHHEE1KPLAQSKHTKHKPVKYIF
VLSPADTNVKAAGWVGHAHEYGAELERMFLSFFTTYTFPHF---DLS---GSAQVKGKGVKAVADTLNAHVDDMPANALSDLHRAHKLVDPPVNF
ISEC1IQVLQSKHPDGDADAGMAKNAPEFLRDMASNRYKELFGFGQ
KLLSCLLLTLaHLPaETPFAHAVSLDKFLASVSTVLTSKR

Figure 1. Alignments of human hemoglobin α (HAHU) (under sequence) to human hemoglobin β (HBHU) and human myoglobin (MYHU) (over sequence) by Murata’s and our algorithms.
CCRF2G
Murata’s algorithm
GSAPPDPVEQKHLITCILCHTDLGK-RNKVGPSLYGVGVRSHGJEPGCYNYSEANIKSG1WTPDLFXY1EHPQ1VPCTKMGYPQFDQPKRADIAYLETL
-----GQVEKGGKIFVQKCAQCHTVGKGGHTGPNLHGLFLGRKTQAPGFSYTDAKNGKTIGEEETMEYLENPKYIPGTRM1FAG1KKTGERADLIAYLKAK
K--
TKE
Our algorithm
GSAPPDPVEQKHLITCILCHTDLGK-RNKVGPSLYGVGVRSHGJEPGCYNYSEANIKSG1WTPDLFXY1EHPQ1VPCTKMGYPQFDQPKRADIAYLETL
-----GQVEKGGKIFVQKCAQCHTVGKGGHTGPNLHGLFLGRKTQAPGFSYTDAKNGKTIGEEETMEYLENPKYIPGTRM1FAG1KKTGERADLIAYLKAK
K---
TKE

CCQF2F
Murata’s algorithm
ADAPTA---FNQ--CKACHSIE-AGKNGVPSLGYGVKVGDPVSYAHLASGMIDEAMLNYPKATIPGKRMGASFGGLRPVEKAV1EYLKTVK--
GQVEKGGKIFVQKCAQCHTVGKGGHTGPNLHGLFLGRKTQAPGFSYTDAKNGKTIGEEETMEYLENPKYIPGTRM1FAG1KKTGERADLIAYLKATKE
Our algorithm
---ADAPTAFNQCKA---CHSIE-AGKNGVPSLGYGVKVGDPVSYAHLASGMIDEAMLNYPKATIPGKRMGASFGGLRPVEKAV1EYLKTVK--
GQVEKGGKIFVQKCAQCHTVGKGGHTGPNLHGLFLGRKTQAPGFSYTDAKNGKTIGEEETMEYLENPKYIPGTRM1FAG1KKTGERADLIAYLKATKE

CCAGC6
Murata’s algorithm
AGEVEKREGMKQGSMALAA1SKGRKPYDAEVKAVATT1STNAKAFDPQFPPGSETGSAAPA1W---ENFD0F6SAAMLADAKYSLPLUSPAGVTAAM--
GQVEK---GK1IFVQKCAQCHTVGKGGHTGPNLHGLFLGRKTQAPGFSYTDANKNGKTIGEEETMEYLENPKYIPGTRM1FAG1KKTGERADL
QTGADCGACHTIQYRLK
IAYLKAK---------KE
Our algorithm
AGEVEKREGMKQGSMALAA1SKGRKPYDAEVKAVATT1STNAKAFDPQFPPGSETGSAAPA1W---ENFD0F6SAAMLADAKYSLPLUSPAGVTAAM--
GQVEK---GK1IFVQKCAQCHTVGKGGHTGPNLHGLFLGRKTQAPGFSYTDANKNGKTIGEEETMEYLENPKYIPGTRM1FAG1KKTGERADL
QTGADCGACHTIQYRLK
IAYLKAK---------KE

Figure 2. Alignments of cytochrome c (CCDG) (under sequence) to Rhodopila globiformis cytochrome C2 (CCRF2G), Rhodospirillum fluvum cytochrome C2 (CCQF2F), and Agrobacterium tumefaciens cytochrome c556 (CCAGC6) (over sequence) by Murata’s and our algorithms.

shown in Figure 4. The FASTA’s optimized scores also quickly decrease from 550 to 25 against Murata’s scores in between 1350 and 1000. On the contrary, the FASTA’s optimized scores do not change to Murata’s scores in between 1000 and 300. It is shown that the FASTA’s optimized scores have no sensitivity in distantly related proteins.

The above results indicate that the similarity scores by our algorithm are more sensitive than the FASTA’s optimized scores in searching distantly related proteins. The computation times by FASTA are two to three times shorter than those by our algorithm. Thus, it is concluded that our system is slower than FASTA in computation times but our scores are more sensitive than those by FASTA.

Our algorithm may be useful for obtaining multiple alignments. The amount of calculation
**Figure 3.** Correlation between Murata's and our similarity scores.

**Figure 4.** Correlation between Murata's similarity scores and the FASTA's optimized scores.
for obtaining a multiple alignment increases exponentially with the number of sequences [8]. As our algorithm works ten times faster than that of Murata, the calculations for obtaining the multiple alignments of three protein sequences with our algorithm are about 1/100 of those of Murata. In our algorithm, the similarity between different types of amino acids is omitted, but this has less influence on obtaining alignments, because the examination of multiple alignments is usually carried out only among similar proteins, which have many amino acid pairs of the same type.

Availability

The source codes of the search and align system using the above algorithms are available from http://www.ulis.ac.jp:9090/nakayama/protein.

References

1) S. B. Needleman and C. D. Wunsch, J. Mol. Biol., 48, 443 (1970).
2) M. Murata, Comput. Chem., 12, 21 (1988).
3) J. P. Dumas and J. Ninio, Nucleic Acids Res., 10, 197 (1982).
4) M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt, in M. O. Dayhoff (ed.), "Atlas of protein sequence and structure," Vol 5 Suppl 3, National Biomed. Res. Foun., Silver Spring, Maryland (1978).
5) W. R. Pearson, Methods enzymol., 183, 63 (1990).
6) D. Eppstein, Z. Galil, R. Giancarlo, and G.F. Italiano, J. ACM, 39, 519 (1992).
7) W. J. Wilbur and D. J. Lipman, Proc. Natl. Acad. Sci. USA, 80, 726 (1983).
8) M. Murata, J. S. Richardson, and J.L. Sussman, Proc. Natl. Acad. Sci. USA, 82, 3073 (1985).