Introducing variable gap penalties into three-sequence alignment for protein sequences

Che-Lun Hung[1], Chun-Yuan Lin[2], Yeh-Ching Chung[1] and Chuan Yi Tang[1]
[1]*Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan ROC*
[2]*Department of Computer Science and Information Engineering, Chang Gung University, Taoyuan, Taiwan, ROC*
allen@sslab.cs.nthu.edu.tw, cyulin@mail.cgu.edu.tw and {ychung,cytang}@cs.nthu.edu.tw

Abstract

The common-use gap penalty strategies, constant penalty and affine gap penalty, have been adopted in the traditional three-sequence alignment algorithm which considers the insertion, deletion and substitution. However, these strategies are not suitable to protein sequence alignments. For the alignment accuracy of protein sequences, the gap penalty is a major determinant. Incorporating protein structure information to vary the gap penalties can lead to more biologically correct alignments. Here, we present an algorithm to find a global and optimal alignment among three protein sequences by using position-specific gap penalties which allow gap penalties to be varied. Thus, residue-dependent information and protein structure information can be applied to the three-sequence alignment. The experimental results show that our algorithm achieves the significant improvement in the accuracy of alignments than the three-sequence alignment algorithm with the affine gap penalty for protein sequences.

Keywords: sequence alignment, three-sequence alignment, variable gap penalties, dynamic programming.

1. Introduction

Three-sequence alignment is the one of multiple sequence alignment methods (MSAs) to align three sequences simultaneously, and it may leads to more accurate and significant similarities than pairwise alignments [7, 13]. Three-sequence alignment can be used to help the biologists to study the DNA homology, the phylogenetic determination and the identification of conserved motifs and protein sequences [8].

Murata *et al.* [13] proposed a three-sequence alignment algorithm, in which a constant gap penalty is applied regardless of the gap length. The affine gap penalty was incorporated into the three-sequence alignment by Gotoh [18]. The space complexity of these algorithms is $O(n^3)$, where n is the maximal length among three sequences. Huang presented an algorithm with the affine gap penalty [25] which is the extension of the algorithm proposed by Myers and Miller [5] by using the Hirschberg's algorithm [4], to reduce the space complexity from $O(n^3)$ to $O(n^2)$. For decreasing the computing time, Lin *et al.* [3] have proposed the parallel three-sequence alignment algorithm with the affine gap penalty to decrease the time complexity from $O(n^3)$ to $O(n^3/p)$, where p is the number of processors. Yue and Tang [6] also presented a three-sequence alignment algorithm with the affine gap penalty and then applied the Divide-and-Conquer technique into this algorithm in [7].

Recently, the three-sequence alignment has been adopted instead of the pairwise alignment in the progressive multiple sequence alignment (progressive MSA). An algorithm, aln3nn [12], applied the three-sequence alignment with the affine gap penalty into the iterative alignment step of progressive MSA. From experimental results, it presented that aln3nn has better results than those by the progressive pair-wise alignment for the RNA alignments, but has bad results for protein sequence alignments. In these algorithms above, gap penalties (constant gap and affine gap penalties) are fixed and may not exactly reflect the biological meanings, especially for protein sequence alignments.

In this paper, we propose a new three-sequence alignment algorithm with variable gap penalties. Initially, we apply the fixed gap opening and the fixed gap extension penalties set by users to calculate the similarity score for any of two sequences among three sequences. Then the gap opening and the gap extension penalties are changed by the weight matrix [14, 21, 23], the similarities of any of two sequences and the lengths of sequences. Finally, we introduce the position-specific gap penalties [9, 10, 24] into the three-sequence alignment. To incorporate the information of known protein structure into existing gaps for the sequence alignment has been indicated to improve the accuracy [15, 17, 20, 26]. Hence, we incorporate this information to vary the gap opening and the gap extension penalties at each residue position in the sequences (position-specific penalties). Therefore, our algorithm can decrease the probability of new gaps inserted in existing protein secondary structure and increase the probability of new gaps inserted in loop regions.

In the experiment tests, we show the comparisons of alignment accuracy between our algorithm and the three-sequence alignment algorithm with the affine gap penalty on the BAliBASE2.0 benchmark [11]. From the experimental results, our algorithm outperforms the three-sequence alignment with the affine gap penalty for protein sequences.

2. Algorithm

2.1 Three-sequence alignment

Let $A = a_1a_2\ldots a_m$, $B = b_1b_2\ldots b_n$ and $C = c_1c_2\ldots c_p$ be three sequences over an alphabet set Σ. The alphabet set contains different symbols representing varied amino acids; for protein sequences are 20 symbols and for DNA sequences are 4 symbols. The symbol "-" denotes "gap" in the sequence alignment. An alignment of three sequences consists of ordered elements in $\Sigma \cup \{-\}$. An example alignment of sequences is illustrated in Figure 1. Each column in the alignment is denoted as an aligned triple and contains one or two gaps which are denoted by 1-gap or 2-gap respectively.

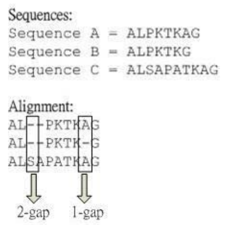

Figure 1. An alignment of three sequences. Each column in the alignment is an aligned triple and "-" denotes a gap.

Let μ and γ be two non-negative constants: μ denotes the gap opening penalty and γ denotes the gap extension penalty. Hence, μ_1 and γ_1 are the gap opening and the gap extension penalties for 1-gap and μ_2 and γ_2 are the gap opening and the gap extension penalties for 2-gap. The $m(x,y)$ is the score to each pair of symbols, where x and y both are in Σ. For protein sequences, the set m is usually the values of the substituted matrix, such as the PAM series [14] and the BLUSM series [21]. The algorithm by using the dynamic programming was proposed to compute an optimal alignment of three sequences by Murata [13]. In the dynamic programming, there are two paths; forward and reverse paths. The alignment scores in the forward path are defined as follows:

$S(i, j, k)$=maximum score for aligning $a_1a_2\ldots a_i$, $b_1b_2\ldots b_j$ and $c_1c_2\ldots c_k$.
$E(i, j, k)$=maximum score for aligning $a_1a_2\ldots a_i$, $b_1b_2\ldots b_j$ and $c_1c_2\ldots c_k$ with b_j gaps.
$F(i, j, k)$=maximum score for aligning $a_1a_2\ldots a_i$, $b_1b_2\ldots b_j$ and $c_1c_2\ldots c_k$ with a_i gaps.
$G(i, j, k)$=maximum score for aligning $a_1a_2\ldots a_i$, $b_1b_2\ldots b_j$ and $c_1c_2\ldots c_k$ with c_k gaps.
$H(i, j, k)$=maximum score for aligning $a_1a_2\ldots a_i$, $b_1b_2\ldots b_j$ and $c_1c_2\ldots c_k$ with b_j and c_k gaps.
$I(i, j, k)$=maximum score for aligning $a_1a_2\ldots a_i$, $b_1b_2\ldots b_j$ and $c_1c_2\ldots c_k$ with a_i and c_k gaps.
$J(i, j, k)$=maximum score for aligning $a_1a_2\ldots a_i$, $b_1b_2\ldots b_j$ and $c_1c_2\ldots c_k$ with a_i and b_j gaps.

The score $S(i, j, k)$ used to compute the score of an optimal alignment of $A_{1,i}$, $B_{1,j}$, and $C_{1,k}$, along with auxiliary scores are given above. The recurrence relations among the auxiliary scores are defined in [18, 25].

In the reverse path, let R, T, U, V, W, X, and Y be the set of the scores that are similar to the scores given above, where R corresponds to S, T to E, U to F, V to G, W to H, X to I and Y to J. For example, $R(i, j, k)$ is the score of an optimal alignment of $A_{i+1,m}$, $B_{j+1,n}$ and $C_{k+1,p}$ for $0 \leqq i \leqq m$, $0 \leqq j \leqq n$ and $0 \leqq k \leqq p$.

The score of an optimal alignment of sequences A, B and C is computed according to the forward and reverse paths. The score is defined as:
$Score = Max\{S(i, j, k) + R(i, j, k), E(i, j, k) + U(i, j, k) + \mu_1, F(i, j, k) + V(i, j, k) + \mu_1, J(i, j, k) + Z(i, j, k) + \mu_2\}$, where $0 \leqq i \leqq m$, $0 \leqq j \leqq n$ and $0 \leqq k \leqq p$.

2.2 Introducing variable gap penalties

The gap penalties computed in the scores given above are fixed. In our algorithm, the fixed gap penalties are modified to the variable gap penalties. The variable gap penalties are calculated by gap penalty functions instead of constant numbers. The gap opening penalty and the gap extension penalty functions are denoted by *GOP* and *GEP*, respectively. Let the gap opening penalty and the gap extension penalty functions be defined as follows:

$GOP_i(x, y)$ returns the gap opening penalty for opening a gap in the sequence i for an entry (x, y) of the score matrices, where x and y are two residue positions in the sequences i and j, respectively.

$GOP_i(x, y, z) = \max[GOP_i(x, y), GOP_i(x, z)]$ return the gap opening penalty for opening a gap in the sequence i for an entry (x, y, z) of the score matrices, where x, y and z are three residue positions in the sequences i, j and k, respectively.

$GOP_{i,j}(x, y, z) = \max[GOP_i(x, z), GOP_j(y, z)]$ returns the gap opening penalty for opening a gap in the sequence i and a gap in the sequence j for an entry (x, y, z) of the score matrices.

$GEP_i(x, y)$ returns the gap extension penalty for extending a gap in the sequence i for an entry (x, y) of the score matrices.

$GEP_i(x, y, z) = \max[GEP_i(x, y), GEP_i(x, z)]$ returns the gap extension penalty for extending a gap in the sequence i for an entry (x, y, z) of the score matrices.

$GEP_{i,j}(x, y, z) = \max[GEP_i(x, z), GEP_j(y, z)]$ returns the gap extension penalty for extending a gap in the sequence i and a gap in the sequence j for an entry (x, y, z) of the score matrices.

Hence, the original scores are modified by replacing the fixed gap penalties with gap penalty functions, and the modified scores are defined as follows:

$$S(i,j,k)=\begin{cases} 0 & \text{if } i=0, j=0 \text{ and } k=0 \\ -\left(GOP_{B,C}(1,0,0)+\sum_{l=1}^{l=i} GEP_{B,C}(l,0,0)\right) & \text{if } i>0, j=0 \text{ and } k=0 \\ -\left(GOP_{A,C}(0,1,0)+\sum_{l=1}^{l=i} GEP_{A,C}(0,l,0)\right) & \text{if } i=0, j>0 \text{ and } k=0 \\ -\left(GOP_{A,B}(0,0,1)+\sum_{l=1}^{l=i} GEP_{A,B}(0,0,l)\right) & \text{if } i=0, j=0 \text{ and } k>0 \\ \max\left[G(i,j,k),H(i,j,k),I(i,j,k)\right] & \text{if } i>0, j>0 \text{ and } k=0 \\ \max\left[E(i,j,k),H(i,j,k),J(i,j,k)\right] & \text{if } i>0, j=0 \text{ and } k>0 \\ \max\left[F(i,j,k),I(i,j,k),J(i,j,k)\right] & \text{if } i=0, j>0 \text{ and } k>0 \\ \max\begin{bmatrix} E(i,j,k),F(i,j,k),G(i,j,k), \\ H(i,j,k),I(i,j,k),J(i,j,k), \\ S(i-1,j-1,k-1)+m(a_i,b_j) \\ +m(a_i,c_k)+m(b_j,c_k) \end{bmatrix} & \text{if } i>0, j>0 \text{ and } k>0 \end{cases}$$

$$E(i,j,k)=\begin{cases} S(i,j,k)-GOP_B(j,k) & \text{if } i=0 \\ S(i,j,k)-GOP_B(i,j) & \text{if } k=0 \\ \max\left[E(i-1,j,k-1),S(i-1,j,k-1)-GOP_B(i,j,k)\right] & \text{if } i>0 \text{ and } k>0 \\ \quad +m(a_i,c_k)-GEP_B(i,j,k) & \end{cases}$$

$$F(i,j,k)=\begin{cases} S(i,j,k)-GOP_A(i,k) & \text{if } j=0 \\ S(i,j,k)-GOP_A(i,j) & \text{if } k=0 \\ \max\left[F(i,j-1,k-1),S(i,j-1,k-1)-GOP_A(i,j,k)\right] & \text{if } j>0 \text{ and } k>0 \\ \quad +m(b_j,c_k)-GEP_A(i,j,k) & \end{cases}$$

$$G(i,j,k)=\begin{cases} S(i,j,k)-GOP_C(j,k) & \text{if } i=0 \\ S(i,j,k)-GOP_C(i,k) & \text{if } j=0 \\ \max\left[G(i-1,j-1,k),S(i-1,j-1,k)-GOP_C(i,j,k)\right] & \text{if } i>0 \text{ and } j>0 \\ \quad +m(a_i,b_j)-GEP_C(i,j,k) & \end{cases}$$

$$H(i,j,k)=\begin{cases} S(i,j,k)-GOP_{B,C}(1,j,k) & \text{if } i=0 \\ \max\left[H(i-1,j,k),S(i-1,j,k)-GOP_{B,C}(i,j,k)\right] & \text{if } i>0 \\ \quad -GEP_{B,C}(i,j,k) & \end{cases}$$

$$I(i,j,k)=\begin{cases} S(i,j,k)-GOP_{A,C}(i,1,k) & \text{if } j=0 \\ \max\left[I(i,j-1,k),S(i,j-1,k)-GOP_{A,C}(i,j,k)\right] & \text{if } j>0 \\ \quad -GEP_{A,C}(i,j,k) & \end{cases}$$

$$J(i,j,k)=\begin{cases} S(i,j,k)-GOP_{A,B}(i,j,1) & \text{if } k=0 \\ \max\left[I(i,j,k-1),S(i,j,k-1)-GOP_{A,B}(i,j,k)\right] & \text{if } k>0 \\ \quad -GEP_{A,B}(i,j,k) & \end{cases}$$

In ClustalW [9], the gap opening penalty (μ) and the gap extension penalty (γ) can be given by user or program initially, and the gap penalties are calculated by the gap penalty schemas automatically to improve the accuracy of alignment. We adopt these schemas to our algorithm, and the schemas are modified to apply to three-sequence alignment. The initial gap penalties are recalculated depending on the following schemas.

Factors for the Gap opening penalty

Dependence on the weight matrix.
The accuracy of the alignment can be improved by using varied weight matrices. The average score for two mismatched residues (i.e. off-diagonal values in the weight matrix) of weight matrix is used as the gap weight for the gap opening penalty.

Denpence on the similarity of the sequences.
For alignments, the relation of the gap opening penalty and the similarity of sequences are in the direct proportion; increasing the penalty for strong similarity; decreasing for the divergent on the linear scale.

Dependence on the lengths of the sequences.
The gap opening penalty is increased with the logarithm of the length of the shorter sequence to avoid that grows the alignment score with the increasing sequence length, even with the false alignment.

Let μ_{ij} and γ_{ij} be the gap opening and the gap extension penalties for sequences i and j. Let η_{ij} be the percent identity for a pair of sequences and w be the average score of mismatched residues of the weight matrix. According to the factors described above, the gap opening penalty is calculated as follows:

$$\mu_{ij}=\mu_{ji} = \{\mu + log[\min(len_i,len_j)]\} * w * \eta_{ij} \qquad (1)$$

where len_i and len_j are the lengths of sequences i and j, respectively. Thus, we have

$$\mu_{AB} = \mu_{BA} = \{\mu + log[\min(m,n)]\} * w * \eta_{AB},$$
$$\mu_{AC} = \mu_{CA} = \{\mu + log[\min(m,p)]\} * w * \eta_{AC},$$
$$\text{and } \mu_{BC} = \mu_{CB} = \{\mu + log[\min(n,p)]\} * w * \eta_{BC},$$

where m, n, and p are the lengths of sequences A, B and C, respectively.

Factor for the Gap extension penalty

Dependence on the difference in the lengths of the sequences.
To avoid producing many long gaps on the sequence which is much shorter than another, the gap extension penalty has to be increased. The initial gap extension penalty is calculated as follow:

$\gamma_{ij} = \gamma_{ji} = \{\gamma + log[\min(len_i, len_j)]\}$ (2)
where len_i and len_j are the lengths of sequences i and j, respectively.
Then, the modified gap extension penalties are

$$\gamma_{AB} = \gamma_{BA} = \{\gamma + log[\min(m,n)]\},$$
$$\gamma_{AC} = \gamma_{CA} = \{\gamma + log[\min(m,p)]\},$$
$$\text{and } \gamma_{BC} = \gamma_{CB} = \{\gamma + log[\min(n,p)]\},$$

where m, n, and p are the lengths of sequences A, B and C, respectively.

In the three-sequence alignment with the affine gap penalty, the initial gap opening and the gap extension penalties are applied equally at every position in the sequence. However, including the information about known protein structure and existing gaps can lead to alignments that are more biologically correct [22]. Hence, we introduce the position-specific gap penalties into our algorithm. The gap penalty tables are used to record the gap opening and the gap extension penalties at each residue position along the lengths of sequences.

The local gap penalty modification rules, applied in a hierarchical manner, are used to calculate penalty tables at each position. The details of rules are given below. First, to avoid that gaps are too close, if there is no gap at a position, but the position is within eight residues of an existing gap, then the gap opening penalty is increased. Second, for alignments of hydrophilic stretches, the gap opening penalty is decreased at a position within a run of five hydrophilic residues. A hydrophilic stretch is formed possibly by any run of 5 hydrophilic residues. The hydrophilic residues that may be set by the users but be conservatively set to D, E, G, K, N, Q, P, R or S by the defaults. The loop regions are usually indicated in these runs. Moreover, if there are no runs of hydrophilic residues or gaps, then the gap opening penalty is modified by using a table of residue-specific gap propensities [22].

Let $GOT_{ij,i}$ be the gap opening penalty table which records the penalty along the length of sequence i for each pair of sequences; sequences i and j. Let $GET_{ij,i}$ be the gap extension penalty table. The gap penalties modified by rules are shown as follows:
Initially, $GOT_{ij,i}[x] = \mu_{ij}$ and $GET_{ij,i}[x] = \gamma_{ij}$ present penalties at x-th residue position.

Increased gap penalties near existing gaps.
$GOT_{ij,i}[x] = \mu_{ij} * \{2+[(8\text{-distance for previous gap})*2]/8\}$ (3)
Reduced gap penalties in hydrophilic stretches.
$GOT_{ij,i}[x] = \mu_{ij} * 0.5$ (4)
If a hydrophilic residue is at the xth residue position within a hydrophilick stretch.
Residue-specific penalties.
$GOT_{ij,i}[x] = \mu_{ij} * R[S_x]$ (5)
Where $R[S_x]$ is the value of the residue in the Residue-specific table R, and the residue is located on the xth position of sequence S.

Recalling the *GOP* and *GEP* functions, these functions are calculated according to equations (3), (4) and (5).
$GOP_i(x, y) = GOT_{ij,i}[x] + GOT_{ij,j}[y]$ (6)
$GEP_i(x, y) = GET_{ij,j}[y]$ (7)

3. Experiments

The benchmark BAliBASE2.0 [11] is a database of manually-refined MSAs specifically designed for the evaluation and comparison of MSA programs. The database has eight reference data sets, and the reference data set 1 is the largest subset with 83 data sets, comprising almost 60% of the benchmark. Each set in reference sets contains more than three sequences. For our experiments, three sequences were randomly selected from the reference set 1 to the reference set 2. The total of 250 test sets is used for the experiments. All of the calculations were performed on a Linux operation system (AMD opteron 250 2.4G with 512MB of memory). We did not adopt other reference sets (3-8) because the differences of percent identity of pair of sequences are big. For example, the maximum percent identity is 81% and the minimum percent identity is 1% in the sub set of the reference set 3. In the experiments, we want to compare the resulting alignments with three different percent identity ranges: <=25%, 20%-40% and >35%. Hence, reference sets 3-8 are hard to be used in our experiments.

In our experiments, the accuracy of the alignment is evaluated by two different scoring functions; sum of pair and total column score [11]. The sum of pair (SP score) indicates the number of correctly aligned residue pairs found in the test alignment divided by the total number of aligned residue pairs in the reference alignment. The total column score (TC score) indicates the number of correctly aligned columns found in the test alignment divided by the total number of aligned columns in the reference alignment. In general, the average scores of SP and TC are used to compare the resulting alignments [1, 2, 12, 19]. In the experiments, our algorithm is denoted by "Three-Align*" and the three-sequence alignment with the affine gap penalty is denoted by "Three Align".

Table 1 presents the comparison of SP and TC scores between the three-sequence alignment algorithm with the affine gap penalty [18] and our algorithm. In the experiments, the following gap opening and the gap extension penalties were used: 1-gap opening penalty = 2-gap opening penalty = 10 and 1-gap extension penalty = 2-gap extension penalty = 2. From Table 1, it is obvious that our algorithm improves almost 28% on average and also obtains better alignments in 75% of the test sets.

In the following experiments, all of 250 test sets are classified into three percent identity ranges: <=25%, 20%-40% and >35%. In the Figure 2, it presents the comparison of different identity percentages with SP scores. Our algorithm achieves 40% improvement than the three-sequence alignment with the affine gap penalty in the range <=25%; 36% improvement in the range 20%-40% and 19% improvement in the range >35%. Our algorithm also obtain better alignments approximates 65%, 76%, and 87% of test sets in three identity ranges respectively.

Figure 3 shows the comparison of different identity percentages with TC scores. Our algorithm achieves 50% improvement than the three-sequence alignment with the affine gap penalty in the range <=25%, 45% improvement in the range 20%-40% and 26% improvement in the range >35%.

Figure 4 shows the comparison of performance between our algorithm and the three-sequence alignment with the affine gap penalty. In our algorithm, we need to calculate the gap penalty matrixes for the variable gap penalties. There are three matrixes used to record the variable gap penalties at each residue position, and the time complexity for building three matrixes is $O(3n^2)$. Hence for our algorithm, the time complexity is $O(n^3 + 3n^2)$, and it can be denoted as $O(n^3)$. The time complexity of the three-sequence alignment with the affine gap penalty is $O(n^3)$. Actually, Figure 4 shows that sequence length will affect the computing time of calculation of gap penalty matrixes.

4. Conclusion

We have presented a new three-sequence alignment algorithm including variable gap penalties and the evidence that it creates alignments with the average accuracy superior to the three-sequence alignment algorithm with the affine gap penalty.

The variable gap penalties introduced here is adopted to replace the fixed gap penalties to achieve better approximate to the block-like behavior of protein alignments. It has been shown that the accuracy of the sequence alignment is improved by incorporating with protein structure information. For example, the equation (5) uses loop propensities, derived by Pascarella and Argos [22], as the scale factors to calculate the penalty at each residue position. Without such structural information, the positioning of gaps in an alignment may influence biological correctness. Although some new gap penalty functions have been proposed to improve the accuracy of protein sequence alignments, these functions are implemented for pairwise alignment and the run time will be increased by calculating gap penalties. Hence these functions cannot be used easily in the three-sequence alignment.

Recently, some progressive MSA methods adopted the three-sequence alignment. However, these methods have bad results for protein sequence alignments because of the affine gap penalty. We have incorporated our algorithm into such progressive MSA to observe the resulting alignments and many alignments have been improved. In the future, we envisage applying our algorithm to compare the protein sequences between different species and including more gap penalty functions and know protein structure information to compute alignments.

Table 1. The comparison of SP and TC scores between two three-sequence alignment algorithms.

	Ave	
	SP	TC
Three Align*	0.82	0.76
Three Align	0.64	0.56

Three-Align* is our algorithm and Three-Align is the three-sequence algorithm method with the affine gap penalty. Ave denotes the Average score.

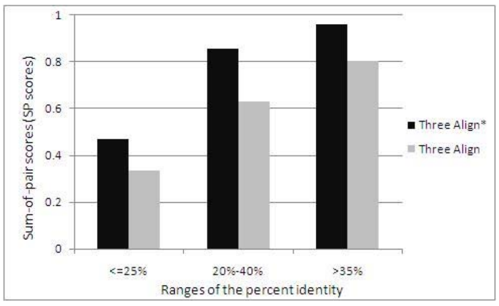

Figure 2. The comparison of three identity percentages ranges with sum-of-pair scores (SP scores).

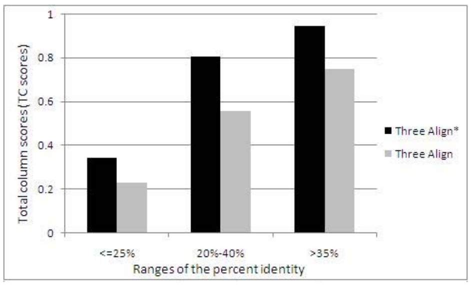

Figure 3. The comparison of three identity percentages ranges with total column scores (TC scores).

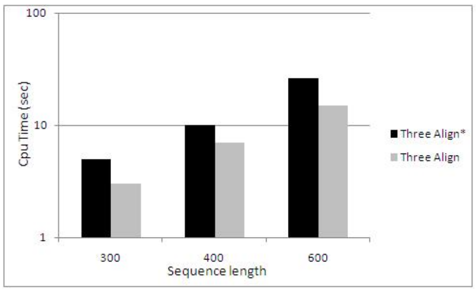

Figure 4. The comparison of performance.

References

[1] C. Grasso and C. Lee, "Combining partial order alignment and progressive multiple sequence alignment increases alignment speed and scalability to very large alignment problems," *Bioinformatics*, vol. 20, 2004, pp. 1546-1556.

[2] C. Notredame, D.G. Higgins and J. Heringa, "T-Coffee: A novel method for fast and accurate multiple sequence alignment," *J. Mol. Biol.*, vol. 302, 2000, pp. 205-217.

[3] CY Lin, CT Huang, YC Chung and CY Tang, "Efficient Parallel Algorithm for Optimal Three-Sequences Alignment," *ICPP*, 2007 (CD-only).

[4] D.S. Hirschberg, "A linear space algorithm for computing maximal common subsequences," *Commun. ACM*, vol. 18, 1975, pp. 341-343.

[5] E.W. Myers and W. Miller, "Optimal alignments in linear space," *Comput. Appl. Biosci.*, vol. 4, 1988, pp. 11-17.

[6] F. Yue, and J. Tang, "Three Sequence Alignment and Ancestor Inference with Affine Gap Costs," *Biocomp*, 2007.

[7] F. Yue and J. Tang, "A Divide-and-Conquer Implementation of Three Sequence Alignment and Ancestor Inference with Affine Gap Costs," *BIBM*, 2007.

[8] H. Makoto, H. Maski, I Masato and T. Tomoyuki, "MASCOT: multiple alignment system for protein sequences based on three-way dynamic programming," *J. Mol. Biol.*, vol. 9, 1993, pp. 161-167.

[9] J.D, Thompson, D.G. Higgins and T.J. Gibson, "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice," *Nucleic Acids Res.*, vol. 22, 1994, pp. 4673-4680.

[10] J. D Thompson, "Introducing variable gap penalties to sequence alignment in linear space," *Bioinformatics*, vol. 11, 1995, pp. 181-186.

[11] J.D. Thompson, F. Plewniak and O. Poch, "BaliBase: a benchmark alignment database for the evaluation of multiple sequence alignment programs," *Bioinformatics*, vol. 15, 1999, pp. 87-88.

[12] K. Matthias K and F.S. Peter FS, "Progressive multiple sequence alignments from triplets," *BMC Bioinformatics*, vol. 8, 2007, 254-266.

[13] M. Murata, J.S. Richardson and J.L. Sussman, "Simultaneous comparison of three protein sequences," *PNAS*, vol. 82, 1985, pp. 3073-3077.

[14] M.O. Dayhoff, R.M. Schwartz and B.C. Orcutt, "A model of evolutionary change in proteins," *Atlas of Protein Sequence and Structure*, vol. 5, 1978, pp. 3345-352.

[15] M.S. Madhusudhan, M, A, Marti-Renom, R. Sanchez and A. Sali, "Variable gap penalty for protein sequence-structure alignment," *Prot. Eng. Des. Sel.*, vol. 11, 2006, pp. 129-133.

[16] M.S. Rosenberg, "Multiple sequence alignment accuracy and evolutionary distance estimation," *BMC Bioinformatics*, vol. 6, 2005, 278-288.

[17] O. Goto, "Alignment of three biological sequences with an efficient traceback procedure," *J. Theor. Biol.*, vol. 121, 1986, pp. 327-337.

[18] O. Gotoh, "Optimal alignment between groups of sequences and its application to multiple sequence alignment," *Comput. Appl. Biosci.*, vol. 9, 1993, pp. 361-370.

[19] R.C. Edgar, "MUSCLE: multiple sequence alignment with high accuracy and high throughput," *Nucleic Acids Res.*, vol. 32, 2004, pp. 1792-1797.

[20] R.F. Smith and T.F. Smith, "Pattern-induced multi-sequence alignment (PIMA) algorithm employing secondary structure dependent gap penalties for use in comparative protein modeling," *Prot. Engng.*, vol. 5, 1992, pp. 35-41.

[21] S. Henikoff and J.G. Henikoff, "Amino acid substitution matrices from protein blocks," *PNAS*, vol. 89, 1992, pp. 10915–10919.

[22] S. Pascarella and P. Argos, "Analysis of insertions/deletions in protein structures," *J Mol Biol*, vol. 224, 1992, pp. 461-471.

[23] T.F. Smith, M.S. Waterman and W.M. Fitch, "Comparative Biosequence Metrics," *J. Mol. Evol.*, vol. 18, 1981, pp. 38-46.

[24] T.G. Dewey, "A sequence Alignment Algorithm with an Arbitrary Gap Penalty Function," *Comput. Biol.*, vol. 8, 2001, pp. 177-190.

[25] X. Huang, "Alignment of three sequences in quadratic space," *Appl. Comput. Review*, vol. 1, 1993, pp. 7-11.

[26] Z.Y Zhu, A. Sali and T.L. Blundell, "A variable gap penalty function and feature weights for protein 3-D structure comparisons," *Prot. Engng.*, vol. 5, 1993, pp. 43-51.