An Efficient Method for Multiple Sequence Alignment

Jin Kim and **Sakti Pramanik**

Department of Computer Science
Michigan State University
East Lansing, MI 48824-1027
{kimj,pramanik}@cps.msu.edu

Abstract

Multiple sequence alignment has been a useful method in the study of molecular evolution and sequence-structure relationships. This paper presents a new method for multiple sequence alignment based on simulated annealing technique. Dynamic programming has been widely used to find an optimal alignment. However, dynamic programming has several limitations to obtain optimal alignment. It requires long computation time and cannot apply certain types of cost functions. We describe detail mechanisms of simulated annealing for multiple sequence alignment problem. It is shown that simulated annealing can be an effective approach to overcome the limitations of dynamic programming in multiple sequence alignment problem.

Key words. computational complexity, dynamic programming, multiple sequence alignment, protein sequence analysis, sequence similarity.

Introduction

Multiple sequence alignment is a useful tool for the search of homology in three or more sequences. It has been helpful in the study of molecular structure, function, and evolution. Pairwise sequence comparisons have been used for sequence similarity. But motifs and other functionally important sites on a sequence may only be identified when a set of sequences are multiple aligned.

Multiple sequence alignment methods can be divided into two different types of algorithms; heuristic algorithms and exhaustive algorithms. Heuristic algorithms (Hogeweg & Hesper 1984; Johnson & Doolittle 1986; Taylor 1987;Barton & Sternberg 1987; Higgins & Sharp 1988; Corpet 1988) try to find out good but not necessarily optimal alignments within a reasonable time. Most of these heuristic algorithms construct a phylogenetic tree for the alignment of the sequences or assign the sequences to a particular order. The sequences are aligned one by one related to the order.

The exhaustive approach (Fredman 1984; Murata, Richardson, & Sussman 1985; Gotoh 1986) based on dynamic programming tries to compare sequences simultaneously. This approach guarantees optimal alignment. Although variations of dynamic programming have been widely used to derive optimal alignments, there are certain limitations.

One important problem in expanding dynamic programming to multiple sequence alignment is its high computational complexity. In pairwise alignment, the computational complexity of dynamic programming is $O(m \cdot n)$ where m, n are the lengths of the sequences. But when dynamic programming is used for multiple sequence alignment, its computational complexity becomes proportional to the product of the lengths of the sequences to be aligned. Therefore, the exponential growth in computational complexity makes dynamic programming impractical for aligning more than three sequences (Fredman 1984; Murata, Richardson, & Sussman 1985; Gotoh 1986). Lipman *et al.* (1989) implemented the Multiple Sequence Alignment (MSA) program to align more than three sequences using dynamic programming. By confining the solution space using heuristic bounds (Carrillo & Lipman 1988), the MSA program can align four to six sequences of length 200-300 residues using rigorous bounds.

Another problem of dynamic programming is its limitation to apply certain cost function in multiple sequence alignment. Altschul (1989) analyzed several types of gap cost and substitution cost for multiple alignments. He pointed out that previously defined gap costs in a multiple alignment were not clearly tied to their substitution costs. He suggested a natural gap cost which was clearly related to its substitution cost. In MSA, quasi-natural gap costs were used instead of natural gap costs because natural gap costs for dynamic programming require impractically long computation time (Altschul 1989). Due to the type of gap costs used, MSA cannot guarantee producing an optimal multiple alignment in some special cases.

Several authors (Lukashin, Engelbrecht, & Brunak 1992;Ishikawa *et al.* 1993) have suggested simulated annealing (SA) as an alternative approach to overcome the limitations of dynamic programming. SA is a good heuristic method to solve combinatorial opti-

mization problems (Kirkpatrick *et al.*, 1983). Ishikawa *et al.*(1993) applied SA to align protein sequences with the same cost function as that used in Gotoh (1986). To reduce the long computation time, they utilized a parallel computer for faster convergence to optimal solution, and discussed temperature parallel algorithm which does not require any temperature scheduling. Lukashin *et al.*(1992) applied SA to human intron sequences with entropy as a cost function.

In this paper, we present details of SA for multiple sequence alignment. To reduce long computation time of traditional SA, several speedup methods are suggested. The SA method for protein sequences is implemented. It is shown that SA can overcome the problem of high computational complexity and the inability to use certain types of cost functions in dynamic programming.

SA for multiple sequence alignment

Simulated annealing

Simulated annealing (SA) was introduced by Kirkpatrick *et al.*, (1983). It is a probabilistic approach that can be used to find a global minimum of a function in combinatorial optimization problems. To apply this algorithm to an optimization problem, a state space $S = \{s_1, \cdots, s_n\}$ and a cost function $C : S \rightarrow R$, where R is the set of real number, should be defined. A real value $C(S)$ should be assigned to each state s. The goal of the optimization problem is to find the optimal state s_{opt} whose score is $min\{s_i \mid 1 \leq i \leq n\}$. Simulated annealing continuously generates a new candidate state s_{new} from a current state $s_{current}$ by applying move sets and acceptance rules. The criteria of the acceptance rules are:

- If $\Delta C \leq 0$, accept a new state s_{new}.
- If $\Delta C > 0$, accept a new state s_{new} with probability $P(\Delta C) = e^{-\Delta C/T}$ where T is a temperature and $\Delta C = C(s_{new}) - C(s_{current})$ is a cost difference.

Probability $P(\Delta C)$ prevents the system from fixation at local minimum. A state $s_{current}$ is called *local minimum* if there is no new state s_{new} in S that is generated from the state $s_{current}$ by applying the single move set and that has a lower cost than that of the $s_{current}$.

Temperature T controls a probability to accept a new candidate state s_{new}. Initially, T starts from a high temperature and after every iteration, T decreases to become zero by applying an annealing schedule. The probability of accepting a new state with a higher cost than that of the current state also decreases as temperature T decreases. If a careful annealing schedule and number of iterations are given, SA converges to a global minimum state s_{opt}. The main disadvantage of SA is its requirement of a large amount of computation time because SA is based on Monte-Carlo methods, which allow for a new candidate state with a higher cost than that of a current state.

Cost function

Each multiple sequence alignment algorithm has its own cost function for the alignment of sequences. To be used in sequence alignment, a cost function C should be explicitly defined as a measure of overall alignment quality.

Altschul (1989) discussed several global cost functions for multiple sequence alignment and suggested SP (sum of pairs) with natural gap costs. SP is the sum of the costs of aligning $n(n-1)/2$ pairs of sequences in an n sequence alignment.

Entropy also can be used as a cost function for multiple sequence alignment. Entropy plays a central role in information theory as measures of information, choice and uncertainty (Shannon 1948). It is considered that an alignment with lower entropy is statistically preferable to an alternative alignment with a higher cost.

Let $S = \{s_1, s_2, \cdots, s_n\}$ be the set of all possible alignments with the same set of sequences. Then the multiple sequence alignment problem is to find the alignment s_i whose cost $C(s_i)$ is smaller than the cost of the other alignment s_j.

One important advantage of SA over dynamic programming is its ability to be performed with any cost function. After applying the move sets to a current alignment, a completely new alignment can be obtained and all the information to be applied to any cost function can be identified. For example, internal gap, external gap, and total number of gaps in a new alignment can be identified after applying move sets to a current alignment. In contrast, any complete alignment cannot be obtained in dynamic programming until all of the computations are finished.

Move Sets

Several move sets can be applied to a current alignment to generate a new candidate alignment. Basically, all the move sets are related to change the positions of the nulls ('-') in the sequences. The basic move sets are as follows.

- *Insertion* $(i, j, k, direction)$: This operation inserts the k number of consecutive nulls from the left/right (direction) of column j in the sequence i.
- *Deletion* $(i, j, k, direction)$: This operation deletes the k consecutive number of nulls from the left/right (direction) of column j where columns $j-\alpha$ through $j+\beta$ $(\alpha, \beta \geq k)$ make a gap where there are consecutive nulls in a sequence.
- *Shuffle* $(i, j, k, direction)$: This operation shuffles the left/right (direction) nulls from the null column j (including null j) in the sequence i and its left/right (direction) k consecutive characters.

Figure 1 shows examples of the move sets.

By modifying these move sets, effective move sets for different type of multiple sequence alignment problem can be obtained. Also move sets can be applied to the

```
A1                  A2
MKQIGGAMGSLA--      MKQIGG--AMGSLA
MKKIGGATGALG--      MKKIGGATGALG--
MK---IGGAMGSLA      MK---IGGAMGSLA

A3                  A4
MKQIGG--AMGSLA      MKQIGGA--MGSLA
MKKIGGATGALG--      MKKIGGATGALG--
MK-IGGAMGSLA--      MK-IGGAMGSLA--
```

Figure 1: Examples of the move sets. (a) original alignment A1. (b) new alignment A2 after $Insertion(1, 6, 2, right)$ to A1. (c) new alignment A3 after $Deletion(3, 5, 2, left)$ to A2. (d) new alignment A4 after $Swap(1, 7, 1, right)$ to A3.

different sequences simultaneously. The parameters i, j and *direction* in the move sets rules may be randomly determined. But k may be determined by certain distribution function, for example uniformly distribution or inverse function related to the size of k. Only experiment can tell which is the best distribution function for k.

Figure 2 shows a sketch of an energy landscape. States are represented along the x axis, with adjacent states being neighbors. The y axis shows the energy. l is a local minimum and g is a global minimum and d is the barrier distance between l and g. The higher barrier distance d prevents l from crossing to g. The time required to cross the barrier d is exponential to d/T where T is a temperature. Carefully designed move

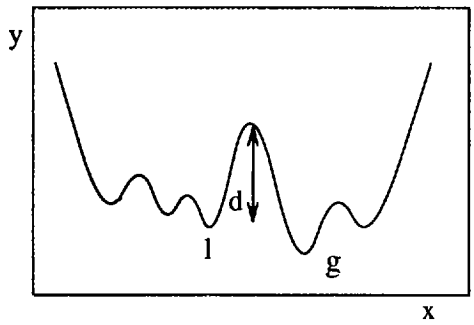

Figure 2: Sketch of energy landscape. l is a local minimum and g is a global minimum. d is a barrier distance between l and g.

sets can lower the barrier distance d and reduce the SA time.

Temperature scheduling

The standard cooling schedule proposed by Kirkpatrick *et al.* (1983) can be written as follows.

$$T_{n+1} = T_n \cdot \gamma \tag{1}$$

where n is the ith annealing step and γ is the constant for reducing the temperature. Let k be the total number of annealing steps and T_E be the final temperature and T_I be the initial temperature. Then the above equation becomes

$$T_f = T_i \cdot \gamma^k \tag{2}$$

From the equation, γ can easily be calculated from T_f, T_i and k as follow.

$$\gamma = (T_f/T_i)^{(1/k)} \tag{3}$$

The starting temperature T_i should be high enough to accept almost any new candidate alignment. At the final temperature T_f, the probability to accept a new candidate alignment with a higher cost ΔC is

$$e^{-\Delta C/T_f} = \epsilon \tag{4}$$

where $0 < \epsilon < 1$. This equation is simplified to

$$T_f = -\Delta C/ln(\epsilon) \tag{5}$$

The minimum cost change, $-\Delta C$, resulting from a move set is 1 when protein sequences are aligned with PAM-250 matrix (Dayhoff 1978) as a substitution cost. Empirically, ϵ^{-1} is set to the total number of iterations k (White 1984). Therefore the final temperature becomes

$$T_f = 1/ln(k) \tag{6}$$

Speedup methods in SA

Heuristic algorithm as the high temperature phase. Simulated annealing is composed of roughly two phases: a high-temperature phase and a low-temperature phase. In the high-temperature phase, SA gives a high probability to all the states with higher costs than that of a current state. This allows any state in the solution set to be a current state. At a lower temperature phase, SA gives a high probability to states with a lower or not much higher temperature than that of a current state. This allows only the states near a current state to be the next state. The high-temperature phase is similar to a random search, and the low-temperature phase is similar to a greedy local search. Rose *et al.* (1986) suggested a good heuristic algorithm as a first phase and a simulated annealing approach as a second phase for fine optimization to the standard-cell-placement algorithm.

In SA for multiple sequence alignment, the same approach can be used. The output alignment generated from the fast heuristic algorithm can be used as the high-temperature phase. Figure 3 shows the annealing curve and different starting points. SA time P can be saved when the system starts from point B which is obtained from the fast heuristic approach instead of point A. It is clear that SA time can be reduced if point A is closer to the optimal point C. When the alignment is obtained from the heuristic approach, the initial temperature should be lower than the initial temperature when traditional SA is applied.

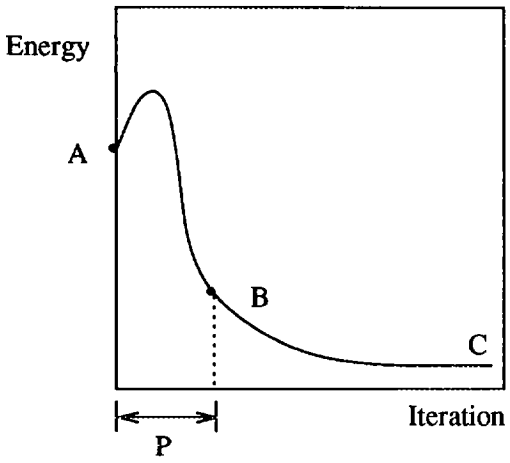

Figure 3: Annealing curve (Energy vs. Iteration). A is the starting point in the traditional SA approach. B is the starting point obtained from the fast heuristic approach. C is the minimal point.

Limitation of lengths for candidate alignments. To apply dynamic programming for aligning n sequences, a fixed amount of computation is required for each cell of the n-dimensional lattice. The total computational time in dynamic programming is proportional to the product of the number of computations for each cell and the size of the n-dimensional lattice. These two factors limit the usage of dynamic programming in multiple sequence alignment. For example, natural gap costs make the number of computations for each cell too large for aligning more than five sequences. The size of the lattice becomes too large for aligning more than three sequences with average protein length of 200-300 residues.

Fickett (1984) suggested a way to reduce the solution space in pairwise alignment. He searched the optimal path within a diagonal band of the two-dimensional matrix as defined by an upper bound of the cost of the optimal path. Carrillo and Lipman (1988) expanded the idea to reduce the solution space for aligning n sequences. They calculated the upper bounds of the alignment cost of each pair of the sequences and confined the bounds. Therefore, they could reduce the computation time by applying dynamic programming to a limited solution space in an n-dimensional lattice.

Execution time in SA can be reduced by confining the length of the candidate alignment. First, initial alignment is obtained by fast heuristic algorithms with the same cost function. Second, only the number of nulls in each sequence is allowed to generate a new candidate alignment. Therefore, only the move sets that change the positions of the nulls are allowed. A column that is composed of all nulls may occur in the candidate alignment. These null columns do not affect the cost of an alignment. Thus, SA can examine the set of alignments, S_i, whose lengths are less than or equal to the length of initial alignment. If the length of the optimal alignment is longer than the length of the initial alignment, SA generates only a near-optimal alignment within the set S_i. By increasing the length of an initial alignment, a larger set of candidate alignments can be examined by SA. To create longer initial alignment a lower gap cost, than the one used in the annealing phase, is applied. However, too much lower gap cost may result in an initial alignment whose length is much longer than that of the optimal alignment. This longer initial alignment requires longer SA time. Therefore a good initial alignment, whose length and cost as close to those of optimal alignment, is crucial for reducing SA time.

Application of SA

Implementation of SA and results for protein sequences

An algorithm called Multiple Sequence Alignment using Simulated Annealing (MSASA) for protein sequences was implemented and compared to MSA based on dynamic programming on the Sun SPARC2. All the parameters in MSASA and MSA were exactly same except gap costs. Natural gap costs were used in MSASA, whereas quasi-natural gap costs were used in MSA. The cost of one gap was 8 in both algorithms. The PAM-250 matrix (Dayhoff 1978) was used for substitution costs. The modified substitution costs (17 minus the values in the PAM-250 matrix) were used in both algorithms. The SP substitution costs in MSA have two options, weighted SP substitution costs and unweighted substitution costs. In the weighted SP substitution costs, weights are applied to the pairwise alignments in order to reduce the effect of the dominance of a set of similar sequences in the multiple sequence alignment. In MSASA, both options could be applied. For easy comparison of the two algorithms, only unweighted SP substitution costs were considered. The experiments were performed on three serine protease families: chymotrypsin, trypsin and elastase.

A heuristic procedure similar to progressive pairwise alignment (Feng & Doolittle 1987) is used to calculate the heuristic bounds in MSA. This heuristic procedure of MSA was also used to generate an initial alignment of MSASA. The number of nulls of each sequence in the initial alignment are allowed to generate a new alignment. Therefore, only the shuffle operation was applied to generate a new candidate alignment in MSASA. In the shuffle operation, the maximum value of parameter k cyclically was changed from 1 to 10 as the number of iteration was increased. The initial temperature T_i was decided by previous experience. The final temperature T_f was determined by equation (6).

Cost and time comparisons

Alignment A1 and A2 in Figure 4 were generated from MSA and MSASA. A2 in figure 4 was generated from

MSASA. The score (2162) of the alignment A2 from MSASA is lower than that (2170) of the alignment A2 from MSA. The difference is due to the different gap costs in MSASA and MSA.

```
A1                                          (2170)
IVGGTNSSWGEWPWQVSLQVKLT-AQRHLCGGSLIGHQWVLTAA
IVNGEEAVPGSWPWQVSLQDKTG---FHFCGGSLINENWVVTAA
IVGGYTCGANTVPYQVSL--NSG---YHFCGGSLINSQWVVSAA
VVGGTEAQRNSWPSQISLQYRSGSSWAHTCGGTLIRQNWVMTAA
VVGGTRAAQGEFPFMVRL--SMG------CGGALYAQDIVLTAA

A2                                          (2162)
IVGGTNSSWGEWPWQVSLQVKLTAQR-HLCGGSLIGHQWVLTAA
IVNGEEAVPGSWPWQVSLQDKTGF---HFCGGSLINENWVVTAA
IVGGYTCGANTVPYQVSL--NSGY---HFCGGSLINSQWVVSAA
VVGGTEAQRNSWPSQISLQYRSGSSWAHTCGGTLIRQNWVMTAA
VVGGTRAAQGEFPFMVRL--SMG------CGGALYAQDIVLTAA
```

Figure 4: Alignment of segments from human plasma kallikrein, bovine chymotrypsin, bovine trypsin, pig elastase, and *Streptomyces griseus* trypsin generated from MSA and MSASA. A1 is the alignment generated from MSA and A2 is generated from SA. The score of A1 is 2170 and the score of A2 is 2162.

Figures 5 and 6 show an example of the alignment of the five sequences generated from MSASA and MSA. These sequences are human plasma kallikrein, bovine chymotrypsin, bovine trypsin, pig elastase, and *Streptomyces griseus* trypsin. The cost of alignment from MSA is 35853 and the cost of alignment from MSASA is 35845, which is lower than MSA's. It took approximately 50 minutes to generate the alignment Figure 6 in MSA. The alignment in Figure 6 was the best alignment selected from the several different runs. Each run took approximately 20 minutes. Quasi-natural gap costs prevent MSA from generating optimal alignment. When a series of nulls with left and right letters is completely imposed on the series of nulls in other sequences, quasi-natural costs count one more gap than natural gap costs do (Altschul, 1989). These additional counts prevent generating optimal alignment from MSA. There are no additional counts in SA because the natural gap costs were used in MSASA.

MSA and MSASA generate the same alignment in some cases. When quasi-natural gap costs are used in both algorithms, both generate the same alignments. And even when natural gap costs are used in MSASA, if there is no completely imposed nulls in the optimal alignment, both algorithms generate the same alignments with rigorous bounds.

When the lengths of the sequences are short and the number of sequences is small, MSA generated optimal alignment faster than MSASA. In MSASA, usually 1 to 5 million iterations, taking 5 to 30 minutes, were enough to get a near-optimal alignment for four to six sequences.

MSA took an impractically long time to align more than six sequences on a personal workstation. Therefore, it could not be directly compared to MSASA for more than six sequences. In MSASA, 1 to 10 million iterations were enough to get a near-optimal solution when aligning up to 10 sequences. Therefore, the running time to align more than six protein sequences in MSASA is more practical than that in MSA.

Discussion

It has been shown that SA is a useful method for multiple sequence alignment. These are the main characteristics of SA.

- Flexibility: As already discussed in the section application of SA, SA can be applied to any cost function. This is because after applying a move set, a complete new alignment can be obtained and any cost function can be applied to this new alignment. Therefore, any constraints or any human knowledge can be incorporated into SA. But its computational complexity prevents dynamic programming from applying certain types of cost functions and constraints.
- Optimality: SA does not guarantee an optimal solution but dynamic programming guarantees an optimal solution with rigorous bounds. The reasons for generating lower costs in the above application is its use of natural-gap costs. In SA, generally longer SA time generates a solution closer to an optimal solution. By applying several speedup methods, this long computation time can be significantly reduced. Hirosawa *et al.* (1993) suggested SA as a refinement tool when long annealing time is required for multiple sequence alignment. They first produced initial alignment by using three way dynamic programming and applied SA for refinement of the initial alignment. With this method they could produce better quality of multiple sequence alignment, with small increases in computational time, than those generated from the heuristic algorithms (Johnson & Doolittle 1986; Taylor 1987;Barton & Sternberg 1987; Higgins & Sharp 1988).

References

Altschul, S. F. 1989. Gap costs for multiple sequence alignment. *J. Theor. Biol.* 138:297–309.

Barton, G. J., and Sternberg, M. J. E. 1987. A startegy for the rapid multiple alignment of protein sequences: confidence levels from tertiary structure comparisons. *J. Molec. Biol.* 198:327–337.

Carrillo, H., and Lipman, D. 1988. The multiple sequence alignment problem in biology. *SIAM J. Appl. Math* 48:1073–1082.

Corpet, F. 1988. Multiple sequence alignment with hierarchical clustering. *Nucl. Acids Res.* 16:10881–10890.

MSA

```
IVGGTNSSWGEWPWQVSLQVKLT-AQRHLCGGSLIGHQWVLTAAHCFDGLPLQDVWRIYSGILNLSDITKDTPFSQIKEIIIHQNYKVSEG--NHDIALI
IVNGEEAVPGSWPWQVSLQDKTG---FHFCGGSLINENWVVTAAHCGVT----TSDVVVAGEFDQGSSSEKIQKLKIAKVFKNSKYNSLTI--NNDITLL
IVGGYTCGANTVPYQVSL--NSG---YHFCGGSLINSQWVVSAAHCYKS-----GIQVRLGEDNINVVEGNEQFISASKSIVHPSYNSNTL--NNDIMLI
VVGGTEAQRNSWPSQISLQYRSGSSWAHTCGGTLIRQNWVMTAAHCVDR---ELTFRVVVGEHNLNQNNGTEQYVGVQKIVVHPYWNTDDVAAGYDIALL
VVGGTRAAQGEFPFMVRL--SMG------CGGALYAQDIVLTAAHCVSG----SGNNTSITATGGVVDLQSAVKVRSTKVLQAPGYNGT----GKDWALI

KLQAPLNYTEFQKPICLPSKGDTSTIYTNCWVTGWGFSK-EKGEIQNILQKVNIPLVTNEECQKR-YQDYKITQRMVCAGYK-EGGKDACKGDSGGPLVC
KISTAASFSQTVSAVCLPSASDDFAAGTTCVTTGWGLTRYTNANTPDRLQQASLPLLSNTNCKK--YWGTKIKDAMICAG---ASGVSSCMGDSGGPLVC
KLKSAASLNSRVASISLPTSCASAG--TQCLISGWGNTKSSGTSYPDVLKCLKAPILSDSSCKSA-YPG-QITSNMFCAGYL-EGGKDSCQGDSGGPVVC
RLAQSVTLNSYVQLGVLPRAGTILANNSPCYITGWGLTR-TNGQLAQTLQQAYLPTVDYAICSSSSYWGSTVKNSMVCAGG--NGVRSGCQGDSGGPLHC
KLAQPINQPTLKIATTTAYNQGTFT------VAGWGANR-EGGSQQRYLLKANVPFVSDAACRSA-YGNELVANEEICAGYPDTGGVDTCQGDSGGPMFR

KHN-GMWRLVGITSWGE--GCARREQPGVYTKVAEYMDWILEKTQSS
KKN-GAWTLVGIVSWGS--STCSTSTPGVYARVTALVNWVQQTLAAN
SGK-----LQGIVSWGS--GCAQKNKPGVYTKVCNYVSWIKQTIASN
LVN-GQYAVHGVTSFVSRLGCNVTRKPTVFTRVSAYISWINNVIASN
KDNADEWIQVGIVSWGY--GCARPGYPGVYTEVSTFASAIASAARTL
```

Figure 5: Alignment of human plasma kallikrein, bovine chymotrypsin, bovine trypsin, pig elastase, and *Streptomyces griseus* trypsin generated from MSA.

SA

```
IVGGTNSSWGEWPWQVSLQVKLTAQR-HLCGGSLIGHQWVLTAAHCFDGLPLQDVWRIYSGILNLSDITKDTPFSQIKEIIIHQNYKVSEGNH--DI
IVNGEEAVPGSWPWQVSLQDKTGF---HFCGGSLINENWVVTAAHCGVT----TSDVVVAGEFDQGSSSEKIQKLKIAKVFKNSKYNSLTINN--DI
IVGGYTCGANTVPYQVSLN--SGY---HFCGGSLINSQWVVSAAHCYKS-----GIQVRLGEDNINVVEGNEQFISASKSIVHPSYNSNTLNN--DI
VVGGTEAQRNSWPSQISLQYRSGSSWAHTCGGTLIRQNWVMTAAHCVDR---ELTFRVVVGEHNLNQNNGTEQYVGVQKIVVHPYWNTDDVAAGYDI
VVGGTRAAQGEFPFMVRLS--MG------CGGALYAQDIVLTAAHCVSG----SGNNTSITATGGVVDLQSAVKVRSTKVLQAPGYNGTG--K--DW

ALIKLQAPLNYTEFQKPICLPSKGDTSTIYTNCWVTGWGFSK-EKGEIQNILQKVNIPLVTNEECQ-KRYQDYKITQRMVCAGY-KEGGKDACKGDS
TLLKISTAASFSQTVSAVCLPSASDDFAAGTTCVTTGWGLTRYTNANTPDRLQQASLPLLSNTNCK--KYWGTKIKDAMICAG---ASGVSSCMGDS
MLIKLKSAASLNSRVASISLPTSCASAG--TQCLISGWGNTKSSGTSYPDVLKCLKAPILSDSSCK-SAYPG-QITSNMFCAGY-LEGGKDSCQGDS
ALLRLAQSVTLNSYVQLGVLPRAGTILANNSPCYITGWGLTR-TNGQLAQTLQQAYLPTVDYAICSSSSYWGSTVKNSMVCAG--GNGVRSGCQGDS
ALIKLAQPINQPTLKIATTTAYNQGTFT------VAGWGANR-EGGSQQRYLLKANVPFVSDAACR-SAYGNELVANEEICAGYPDTGGVDTCQGDS

GGPLVCKHNG-MWRLVGITSWGE--GCARREQPGVYTKVAEYMDWILEKTQSS
GGPLVCKKNG-AWTLVGIVSWGS--STCSTSTPGVYARVTALVNWVQQTLAAN
GGPVVCSGK-----LQGIVSWGS--GCAQKNKPGVYTKVCNYVSWIKQTIASN
GGPLHCLVNG-QYAVHGVTSFVSRLGCNVTRKPTVFTRVSAYVSWIKQTIASN
GGPMFRKDNADEWIQVGIVSWGY--GCARPGYPGVYTEVSTFASAIASAARTL
```

Figure 6: Alignment of human plasma kallikrein, bovine chymotrypsin, bovine trypsin, pig elastase, and *Streptomyces griseus* trypsin generated from MSASA.

Dayhoff, M. O. 1978. A model of evolutionary change in proteins. matrices for detecting distance relationships. In *Atlas of Protein Sequence and Structure*, volume 5 suppl. 3. Dayhoff. M. O.(ed) Washington. DC: National Biomedical Research Foundation. 345–352.

Feng, D. F., and Doolittle, R. F. 1987. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Molec. Evol.* 25:351–360.

Fickett, J. W. 1984. Fast optimal alignment. *Nucl. Acids Res.* 12:175–180.

Fredman, M. L. 1984. Algorithms for computing evolutionary similarity measures with length independent gap penalties. *Bull. Math. Biol.* 46:553–566.

Gotoh, O. 1986. Alignment of three biological sequences with an efficient traceback procedure. *J. Theor. Biol.* 121:327–333.

Higgins, D. G., and Sharp, P. M. 1988. Clustal: a package for performing multiple sequence alignment on a microcomputer. *Gene* 7:237–244.

Hirosawa, M.; Hoshida, M.; Ishikawa, M.; and Toya, T. 1993. Mascot: Multiple alignment system for protein sequences based on three-way dynamic programming. *CABIOS* 9:161–167.

Hogeweg, P., and Hesper, B. 1984. The alignment of sets of sequences and the construction of phyletic trees: an integrated method. *J. Molec. Evol.* 20:175–186.

Ishikawa, M.; Toya, T.; Hoshida, M.; Nitta, K.; Ogiwara, A.; and Kanehisa, M. 1993. Multiple sequence alignment by parallel simulated annealing. *CABIOS* 9:267–273.

Johnson, M., and Doolittle, R. F. 1986. A method for the simulataneous alignment of three or more amino acid sequences. *J. Molec. Evol.* 23:267–287.

Lipman, D. J.; Altschul, S. F.; and Kececioglu, J. D. 1989. A tool for multiple sequence alignment. *Proc. Natl. Acad. Sci. USA.* 86:4412–4415.

Lukashin, A. V.; Engelbrecht, J.; and Brunak, S. 1992. Multiple alignment using simulated annealing: branch point definition in human mRNA splicing tool for multiple sequence alignment. *Nucl. Acids Res.* 20:2511–2516.

Murata, M.; Richardson, J. S.; and Sussman, J. L. 1985. Simultaneous comparison of three protein sequences. In *Proc. Natl. Acad. Sci. USA.*, volume 82, 3073–3077.

Shannon, C. E. 1948. A mathematical theory of communication. *Bell System Techn, J.* 27:379–432,623–656.

Taylor, W. R. 1987. Multiple sequence alignment by a pairwise algorithm. *CABIOS* 3:81–87.

White, S. R. 1984. Concepts of scale in simulated annealing. In *Proc. ICCD*, 646–651.