# A Hybrid Algorithm Based on Artificial Immune System and Hidden Markov Model for Multiple Sequence Alignment

**Hongwei Ge** [1] **Weimin Zhong**[1] **Wenli Du** [1] **Feng Qian** [1] **Lu Wang** [2]

[1]Automation Institute, East China University of Science and Technology, State-Key Laboratory of Chemical Engineering, Shanghai 200237, China
[2]College of Computer Science, Jilin University, Changchun 130012, China

## Abstract

Multiple sequence alignment (MSA) has become an essential tool in the analysis of biologic sequences. In this paper, an artificial immune system (AIS) is proposed to train hidden Markov models (HMMs). Further, an integration algorithm based on the HMM and AIS for the MSA is constructed and a decoding algorithm based on Viterbi algorithm is also proposed. The approach is tested on a set of standard instances taken from the benchmark alignment database, BAliBASE. Numerical results are compared with those obtained by using the Baum-Welch training algorithm. The results show that the proposed algorithm not only improves the alignment abilities, but also reduces the time cost.

**Keywords**: Artificial immune system, hidden Markov model, multiple sequence alignment

## 1. Introduction

Multiple sequence alignment (MSA) is a common task in molecular biology and bioinformatics. From the computational viewpoint, the multiple sequence alignment of proteins or DNA is a very difficult task and it was proved to be *NP*-complete [1]. However, alignment is the most important techniques for discovering and representing similarities between sequences, and can unravel the evolutionary history, critical preserved motifs, and details of the tertiary structure or important clues about function. Therefore, MSA is a recurrent issue of extensive research in bioinformatics and computer science, aiming at finding more efficient algorithms. An optimal sequence alignment can be computed by the methods based on dynamic programming, such as the Needleman-Wunsch algorithm for the global alignment [2], the Smith-Waterman algorithm for the local alignment [3], and so on. However, the running time increases rapidly with the increase of the number of sequences to be aligned. So many heuristics and approximation algorithms have been proposed [4, 5]. With the emerging of new techniques in the field of artificial intelligence, some newly developed methods have been introduced to address these problems in recent years, such as simulated annealing (SA) [6], genetic algorithm (GA) [7]. Besides these methods, a very general form of probabilistic model for sequences of symbols, called a hidden Markov model (HMM), has also been applied to MSA [8]. However, because the MSA is among the members of the class of hard NP-complete problems there remains much room for improvement in current techniques and exploitation of new efficient methods.

In this paper, a novel algorithm for MSA based on artificial immune system (AIS) and hidden Markov model is proposed. The critical and difficult problem for using HMM approach is how to set up a steady and reliable model by a finite training set, moreover, there is no known deterministic algorithm that can guarantee to find the optimally trained HMM with reasonable running time. The most widely used approach to train an HMM is based on statistics and re-estimation, such as Baum-Welch (BW) algorithm [9]. This paper proposes an artificial immune system to train the HMM. The proposed algorithm possesses the randomicity of stochastic optimization algorithms, which can be used to solve the optimization problem for non-linear systems; moreover, the proposed algorithm possesses the adaptive ability that enables the algorithm to solve machine learning problems.

## 2. Hidden Markov Models for Multiple Sequence Alignment

At an abstract level, molecular sequences are long strings of characters over an alphabet of size four for DNA and twenty for protein. Although each base position presents one of four or twenty states, the number of these positions is likely to vary, that is homologous nucleotide sequences or amino acid sequences may differ in length. This sequence length variation has led to the development of the algorithm for multiple sequence alignment, which is used to align these bases so that homologous residues among a set of sequences are aligned together in columns.

Consider $N$ homologous sequences $S_1 = s_{11}s_{12}\cdots s_{1L_1}$, $S_2 = s_{21}s_{22}\cdots s_{2L_2},\cdots$, $S_N = s_{N1}s_{N2}\cdots s_{NL_N}$, where $s_{nl}$ $(n \in [1,N], l \in [1,L_n])$ denotes a single nucleotide or amino acid, collectively called a residue. The set of valid residue types is denoted by $\Sigma$. During the evolutionary process, some residues may stay unchanged while some may change to other types. For simplicity, we regard conservation as matching to itself. In addition, some residues may be lost, or new residues may insert somewhere between or outside the preexisting sites. The final result of the series of mutational events (M for match, I for insert, and D for delete) is conveniently represented by a two-dimensional character matrix $A = (s_{ij})_{N\times M}$, where $s_{ij} \in \Sigma \cup \{-\}$. The null character '–' represents a space. Each row turns out to be a contemporary sequence if all null characters are removed. The gained alignment can be evaluated by different scoring systems. The multiple sequence alignment is combinatorial problem in nature because there are exponentially many ways of inserting spaces to form an alignment.

A hidden Markov model is a statistical model. Mathematically, an HMM is defined by specifying: a set of states $W = \{w_1, w_2, \cdots, w_n\}$, an alphabet $O = \{o_1, o_2, \cdots o_m\}$, transition probability matrix $(a_{ij})_{n\times n}$, $a_{ij} = P(w_j \mid w_i)$, $(1 \le i, j \le n)$, which represents the transition probability from state $w_i$ to state $w_j$; emission probability matrix $(b_{jk})_{n\times m}$, $b_{jk} = b_j(o_k) = P(o_k \mid w_j)$, $(1 \le j \le n, 1 \le k \le m)$, which represents the emission probability of the letter $o_k$ in the state $w_j$. The sum of the probabilities of all transitions going out of a state is 1, and the sum of all emission probabilities in each state is 1. A letter from the alphabet in a state is generated according to the emission probability, then a new state is chosen according to the transition probability and the cycle continues. An HMM can be represented by a graph in which the nodes correspond to the states, and arrows connect nodes for which there is a non-zero transition probability. An HMM for multiple sequence alignment is a state chain including matching (M), insertion (I) and deletion (D) in nature. The HMM structure used in this paper is the standard topology for the MSA problem as shown in Figure 1, in which the length of the HMM is 3.

When applying HMMs to MSA, the sequence of observables is given in the form of some unaligned sequences of amino acids. Different sequences are generated with different probability. A sequence $S$ is generated with probability $P(S \mid \lambda)$, where $\lambda$ is the parameter set including all the transition and emission probabilities and determines an HMM model. The goal is to find the model parameter set $\lambda^*$ such that it maximizes the probability $\prod_{k=1,2,\cdots,K} P(S_k \mid \lambda^*)$, where $S_1,\cdots,S_K$ are the sequences we try to align. The procedure of finding $\lambda^*$ can be thought as training an HMM using the training set $\{S_1,\cdots,S_K\}$. The traditional training algorithm is based on the method of expectation maximization or generalized expectation maximization, such as Baum-Welch algorithm [9]. The core idea in these training algorithms is to update probability parameters using the way of recursion. The computation cost is enormous, and the computation complexity is $O(KN^2)$ at each training step, where $K$ is the number of the sequences and $N$ is the length of the HMM. These training algorithms are only guaranteed to converge to a local maximum in the parameter space. The main problem is that in most applications of interest the optimization surface is very complex and there are many local maxima which trap the model and prevent it from reaching a global maximum. Besides, Baum-Welch algorithm does not have the parameter for learning step, so it is easy to be affected by some isolated samples. In this paper, a vaccination-based artificial immune system is proposed to train HMM. It is prone to avoid getting into local maxima, and the most of the computing time is spent on the evaluation of candidate solutions.

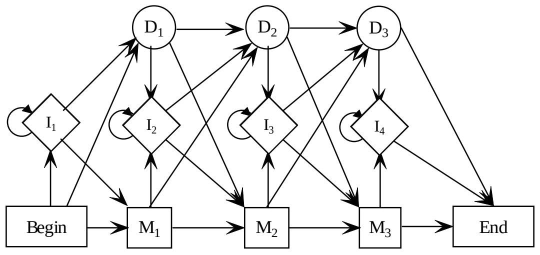

Fig. 1: An HMM structure for MSA.

## 3. AIS-based HMM Learning Algorithm

Artificial immune system is inspired by theoretical immunology and observed immune functions, principles, and models. It is a new artificial intelligence technique which is beginning to mature through the collaborative effort of many interdisciplinary researchers [10]. The immune system is a naturally occurring event-response system which can quickly adapt to changing situations. The operative mechanisms of immune systems are very efficient from a computational standpoint. The AIS has appeared to offer powerful and robust information processing capabilities for solving complex problems

[11-14]. However, to the best of out knowledge, the use of artificial immune systems for training the HMM has been not found in the existing literatures. In this research, an HMM learning algorithm is proposed based on a novel artificial immune system. The proposed HMM learning algorithm is built on the principles of clonal selection, affinity maturation and the abilities of learning and memory.

## 3.1. Clonal Selection

The principle of clonal selection describes how the adaptive immune system reacts to antigens and improves its capability of recognizing and eliminating them. In the AIS algorithm, the ideas embody the following aspects: reproducing the antibody having high affinity with the antigen, restraining the antibody with too higher consistency and eliminating the antibody with lower affinity, getting solution with high diversity. Thus the clonal selection probability is direct proportion to the antibody affinity and inverse proportion to the antibody consistency. Reproducing the antibody with higher affinity can speed up the convergence of the algorithm; on the other hand, restraining the antibody with too higher consistency can avoid the emergence of the prematurity.

In this paper, the antibody affinity degree with the antigen is evaluated by a modified version of the sum-of-pairs function [15], which is discussed in Section 4. The antibody consistency is computed by the following formulation:

$$Density(c_i) = \frac{1}{\rho(c_i)} = \frac{1}{\sum_{j=1}^{N} \left| f(c_i) - f(c_j) \right|} \quad (1)$$

Where $f(c_i)$ and $f(c_j)$ are the affinity of antibody $c_i$ and the affinity of antibody respectively. $N$ is the scale of the antibody population.

The clonal selection probability $P_s(c_i)$ of antibody $c_i$ is defined by

$$P_s(c_i) = \frac{\rho(c_i)}{\sum_{i=1}^{N} \rho(c_i)} = \frac{\sum_{j=1}^{N} \left| f(c_i) - f(c_j) \right|}{\sum_{i=1}^{N}\sum_{j=1}^{N} \left| f(c_i) - f(c_j) \right|} \quad (2)$$

## 3.2. Mutation

In immune systems, random changes take place in the variable region genes of antibody molecules. These random changes are called mutation. Mutations cause structurally different antibody molecules.

Let vector $X = (x_1, x_2, \cdots, x_i, \cdots, x_n)$ denote the antibody to be used for mutation, and vector $Z = (z_1, z_2, \cdots, z_i, \cdots, z_n)$ denote the antibody generated by the mutation operator. $Z$ can be generated by the following rules:

$$z_i = \begin{cases} x_i + \delta_i & if \ \mathrm{mod}(\theta, 2) = 0 \\ x_i - \delta_i & if \ \mathrm{mod}(\theta, 2) = 1 \end{cases} \quad i = (1, 2, \cdots n) \quad (3)$$

Where $\delta_i$ is an adjustment factor of mutation, which is a real number generated randomly in the interval [0, $(b_i - a_i)/2$ ], and $\theta$ is a nonnegative integer generated randomly. If the antibody generated by the above rules is out of the domain of definition, an adjustment is made for $\delta_i$ by the method of dichotomy until the antibody satisfies the given constraint.

## 3.3. Vaccination

In this section, a vaccination model based on immune systems is designed to improve the AIS performance. Similarly to the techniques of the vaccine inoculation, we take full advantage of some characteristic information and knowledge in the process of solving problem to distill vaccines, and then vaccinate some antibodies. After each given generation we distill vaccines from the best antibody and vaccinate some antibodies by the adaptive vaccination probability. The process is designed as follows.

Let $P_v$ be the vaccination probability, and relatively smaller $P_v$ should be adopted for those better antibodies whose affinity values are higher than the average affinity of the population, whereas a relatively larger $P_v$ should be adopted for those antibodies whose affinity values are lower than the average affinity. Let $\bar{f} / f_i = \mu$ . The adaptive vaccination probability $P_v$ can be expressed as

$$p_v = \frac{f_g - f_i}{f_g - \mu \bar{f}}, \quad \text{if } \mu \le 1; \quad P_v = 1, \quad \text{if } \mu > 1. \quad (4)$$

Let the best antibody $P_g = (p_{g1},\ p_{g2}, \ldots,\ p_{gj}, \cdots, p_{gD})$, the vaccinated antibody $X_i = (x_{i1},\ x_{i2}, \ldots x_{ij}, \cdots, x_{iD})$ , $\forall j$ , assume that $p_{gj} \le x_{ij}$ , otherwise exchange $p_{gj}$ with $x_{ij}$ . Let vectors $A$= $(a_1, a_2, \cdots, a_j, \cdots, a_D)$ and $B$= $(b_1, b_2, \cdots, b_j, \cdots, b_D)$ define the lower and upper bounds of the problem domain. For arbitrary antibody $X_k$ , $a_j \le x_{kj} \le b_j$ , $j = (1, 2, \cdots, D)$ . Define

$$p^*_{gj} = x_{ij} + \frac{(x_{ij} - p_{gj})(b_j - x_{ij})}{x_{ij} - a_j} \quad (5)$$

$$x^*_{ij} = p_{gj} - \frac{(x_{ij} - p_{gj})(p_{gj} - a_j)}{b_j - p_{gj}} \quad (6)$$

It is easy to prove that

$$a_j \le x^*_{ij} \le p_{gj} \le x_{ij} \le p^*_{gj} \le b_j . \quad (7)$$

Let vector $Z_i = (z_{i1}, z_{i2}, \cdots, z_{ij} \cdots, z_{iD})$ denote the antibody generated by the vaccination operation. It can be obtained by the following rules:

$$z_{ij} = \begin{cases} \alpha a_j + (1-\alpha) p_{gj}, if \bmod(\theta,4) = 0 \\ Rom(\alpha p_{gj} + (1-\alpha) x_{ij}, \alpha p_{gj} + (1-\alpha) x^*_{ij}), \quad if \bmod(\theta,4) = 1 \\ Rom(\alpha x_{ij} + (1-\alpha) p_{gj}, \alpha x_{ij} + (1-\alpha) p^*_{gj}), \quad if \bmod(\theta,4) = 2 \\ \alpha x_{ij} + (1-\alpha) b_j, if \bmod(\theta,4) = 3 \end{cases}$$

$$j = (1,2,\cdots,D) \tag{8}$$

Where $\alpha$ is a real number generated randomly in the interval [0, 1], and $\theta$ is a nonnegative integer generated randomly. $Rom(x,y)$ is a random selection function, which represents select one randomly from $x$ and $y$.

It is easy to obtain the following conclusions. When $\bmod(\theta,4) = 0$, the generated $z_i$ satisfies that $a_i \le z_i \le x_i$, namely, probabilities in HMM change in the direction of reduction with a tendency towards the lower bound of the domain of definition. When $\bmod(\theta,4) = 3$, $z_i$ satisfies that $y_i \le z_i \le b_i$, namely, probabilities change in the direction of increase with a tendency towards the upper bound of the domain. When $\bmod(\theta,4) = 1$ or $\bmod(\theta,4) = 2$, $z_i$ satisfies that $a_i \le z_i \le b_i$, namely, the changes of probabilities oscillate in the whole domain. Finally, the immune test is performed. If the produced antibody is better than the original antibody, replace the original by the produced.

## 3.4. An AIS-based HMM Learning Algorithm

The brief outline of the proposed HMM learning algorithm based on the artificial immune system can be described as follows:

Step 1. Initialize $N$ antibodies as an initial population, $N$ is the scale of the population.

Step 2. Calculate and adjust the affinity degree of each antibody in the current antibody population.

Step 3. Select *m* antibodies from the population by the clonal selection model and clone them to a clonal library.

Step 4. Perform the mutation operation for each of the antibodies in the clone library.

Step 5. Select *n* antibodies from the clonal library according to the probability $P_v$ and perform the vaccination operation for the selected antibodies.

Step 6. Replace the worst *s* antibodies in the population by the best *s* antibodies from the clonal library.

Step 7. Stop if the termination condition is satisfied, else repeat Step 2-Step 7.

## 4. Multiple Sequence Alignment Based on AIS and HMM

In this section, an integration algorithm based on the hidden Markov model and artificial immune system for the multiple sequence alignment is constructed. Besides, a modified decoding algorithm based on Viterbi algorithm [16] is also proposed to address the MSA problem.

### 4.1. The outline of MSA Based on AIS and HMM

The ordered components in an antibody molecule are defined as all the transition probabilities and emission probabilities when using AIS to train HMM for MSA. For the HMM profile shown in Fig.1, let *N* be the predetermined length of the HMM, which is taken as the average length of the sequences in the alignment problem, and |A| denotes the size of the alphabet. Without regard to the “begin” state and the “end” state, an antibody molecule is a permutation with real encoding of 9*N*+3 transition probabilities and (2*N*+1)|A| emission probabilities, which spans a search space of (2*N*+1)|A|+9*N*+3 dimensions. When *N* is large enough, the number of the probability parameters is 49*N* for protein models and 17*N* for DNA models. So an antibody corresponds to a determined HMM model, and the evaluation of an antibody is also the evaluation for its corresponding HMM model. Before evaluating an antibody all the probability parameters need to be normalized to satisfy that the sum of the probabilities of all transitions going out of a state is 1, and the sum of all emission probabilities in each state is 1. Then the most probable state sequence path for each sequence can be gained by using the proposed decoding algorithm based on Viterbi algorithm. Afterwards, the sequences can be aligned according to the specific states in their paths. Finally, a gained alignment can be evaluated by a scoring system.

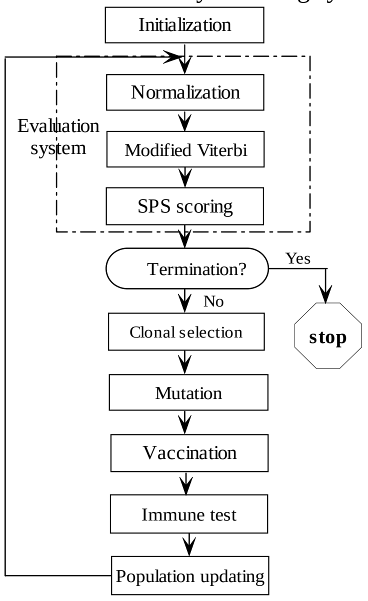

Fig. 2: The flow chart of the algorithm for MSA

Thus far, the proposed algorithm for MSA can be described by the flow chart shown in Figure 2.

### 4.2. Modified Viterbi Decoding Algorithm

In this section, a modified Viterbi algorithm is proposed for decoding the antibody molecule. The modified algorithm can store the current decoded character at each step. Denote the match and insertion states as state 1 and state 2, deletion state as state 3, respectively. In the Viterbi algorithm, define

$$\delta_i(t) = \max_{\pi_i(t)} P(\pi_i(t) \mid \lambda) . \qquad (9)$$

where $\pi_i(t)$ is the prefix path to observe the partial sequence $X_1 \cdots X_t$ up to time $t$ and state $i$ at time $t$ given the model $\lambda$. $\delta_i(t)$ is the probability relative to the most probable path. The variable is updated by the transmission mechanism similar to the forward algorithm. In the decoding process, for the match and insertion states, the computing methods of $\delta_i(t)$ are uniform, while $\delta_i(t)$ needs to be computed separately for the deletion state, because the deletion state does not generate characters. The brief outline of the algorithm is as follows:

Step 1: Initialize

$$\delta_1(i) = \pi_i b_i(O_1) , \qquad (10)$$

$$\Psi_1(i) = \{0,1\} \quad (1 \le i \le 2) , \qquad (11)$$

$$\Psi_1(3) = \{0,0\} . \qquad (12)$$

Step 2: Compute

$$\delta_t(j) = \max_i[\delta_{t-1}(i)a_{ij}]b_j(O_{\Psi_{t-1}(i)[2]+1}), \quad (2 \le t \le T, 1 \le j \le 2) \qquad (13)$$

$$\Psi_t(j) = \{\arg\max_i[\delta_{t-1}(i)a_{ij}], O_{\Psi_{t-1}(i)[2]+1}\}, \quad (2 \le t \le T, 1 \le j \le 2) \qquad (14)$$

$$\delta_t(j) = \max_i[\delta_{t-1}(i)a_{ij}], \quad (2 \le t \le T, j = 3) \qquad (15)$$

$$\Psi_t(j) = \{\arg\max_i[\delta_{t-1}(i)a_{ij}], O_{\Psi_{t-1}(i)[2]}\}, \quad (2 \le t \le T, j = 3) \qquad (16)$$

Where $\Psi_t(j)$ stores the current decoded character and the parameter *i* that makes $\delta_{t-1}(i)a_{ij}$ maximize.

Step 3: Denote

$$P^* = \max_i[\delta_T(i)] \qquad (17)$$

$$X_T^* = \arg\max_i[\delta_T(i)] \qquad (18)$$

Step 4: Compute

$$X_t^* = \psi_{t+1}(X_{t+1}^*) \quad (t = T-1, T-2, \cdots, 1) . \qquad (19)$$

### 4.3. Scoring System

The alignment score, namely the affinity of an antibody, is evaluated by a scoring system. In this paper, a modified version of the sum-of-pairs function [13] is adopted. Given a test alignment of *N* sequences consisting of *M* columns, the *i*th column of the alignment is denoted by $A_{i1}, A_{i2}, \cdots, A_{iN}$ . For each pair of residues $A_{ij}$ and $A_{ik}$ , define $p_{ijk}$ such that $p_{ijk}$ =1 if residues $A_{ij}$ and $A_{ik}$ are aligned with each other in the reference alignment, otherwise $p_{ijk}$ =0. The score $S_i$ for the *i*th column is defined as:

$$S_i = \sum_{j=1, j\ne k}^{N} \sum_{k=1}^{N} p_{ijk} \qquad (20)$$

Then the *SPS* for the alignment is:

$$SPS = \sum_{i=1}^{M} S_i \Big/ \sum_{i=1}^{M_r} S_{ri} \qquad (21)$$

where $M_r$ is the number of columns in the reference alignment and $S_{ri}$ is the score $S_i$ for the *i*th column in the reference alignment.

## 5. Numerical Simulation Results

The performance of the proposed algorithm for MSA is examined by the benchmark problems from the BAliBASE alignment database. BaliBASE is available on the World Wide Web at http://bips.u-strasbg.fr/fr/Products/Databases/BAliBASE2. The database contains reasonably high-quality, well-documented alignments that have been confirmed using a variety of programs and by manual verification. The first version of BAliBASE contains 142 reference alignments, with a total of more than 1000 sequences. It is divided in five hierarchical reference sets by the length and similarity of the sequences in the core blocks and by the presence of insertions and N/C-terminal extensions. We select a total of twelve sequence sets from the reference sets, which are listed in Table 1. Numerical experiments are performed on a PC with Pentium IV 2.8 GHz processor and 512 MB memory. Some parameters in the simulated experiment are taken as: $N$ =50, $m$ =30, $n$ =10, and the iterations are taken as 3000. Numerical simulated results are compared with those obtained by using the Baum-Welch training algorithm and listed in Table 1. The contents of the table include the name of each test problem (Name), the number of sequences (*N*), the length of sequences (LSEQ(m, n), m-minimum, n-maximum), the percentage sequence identity (Id%(a, b, c), a- average, b- minimum, c-maximum), the results gained by using the Baum-Welch algorithm (BW), the results gained by using AIS training algorithm (AIS), and the time cost (Time). It is worth mentioning that the results listed in the table is the best obtained from five executions.

| Name | N | LSEQ(m, n) | Id%(a,b,c) | BW | Time(s) | AIS | Time(s) |
|---|---|---|---|---|---|---|---|
| 1aboA | 5 | (48,74) | <25%(18,14,26) | 0.622 | 121.28 | 0.646 | 71.23 |
| 451c | 5 | (70,87) | 20-40%(27,23,34) | 0.321 | 159.73 | 0.538 | 97.75 |
| 9rnt | 5 | (96,103) | >35%(57,51,65) | 0.783 | 178.78 | 0.804 | 116.15 |
| kinase | 5 | (260,273) | <25%(20,17,25) | 0.308 | 437.65 | 0.399 | 202.33 |
| 2cba | 5 | (237,259) | 20-40%(26,22,33) | 0.653 | 408.63 | 0.761 | 234.71 |
| 1ppn | 5 | (212,220) | >35%(46,41,59) | 0.605 | 383.86 | 0.623 | 178.44 |
| 2myr | 4 | (340,474) | <25%(16,13,25) | 0.236 | 589.71 | 0.385 | 209.08 |
| 1eft | 4 | (334,405) | 20-40%(30,25,35) | 0.728 | 512.34 | 0.739 | 227.61 |
| 1taq | 5 | (806,928) | >35%(40,35,50) | 0.747 | 4278.93 | 0.817 | 1177.83 |
| 1ubi | 17 | (76,97) | -(30,8,71) | 0.267 | 853.42 | 0.393 | 481.30 |
| kinase | 18 | (257,287) | -(32,17,73) | 0.186 | 2119.87 | 0.270 | 814.76 |
| 1idy | 27 | (49,60) | -(19,1,81) | 0.295 | 996.06 | 0.346 | 490.27 |

Table 1: The benchmark alignment problems and simulation results.

From the table it can be seen that each score of the twelve alignment results gained by using the AIS is higher than that using the Baum-Welch algorithm. The average score gained by using the Baum-Welch algorithm is 0.479, whereas that gained by using the AIS is 0.560, which is about 1.2 times of the Baum-Welch algorithm. On the other hand, the time cost of the proposed algorithm is much less than using the Baum-Welch algorithm. So it could be concluded that using the AIS to train HMM for solving MSA is superior to the Baum-Welch algorithm. Figure 3 gives a group of convergence curves for instance laboA.

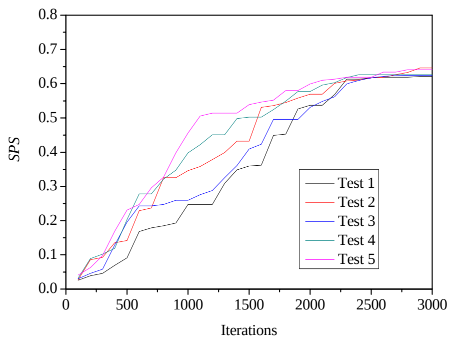

Fig. 3: A group of convergence curves for instance laboA

## 6. Conclusions

We propose an artificial immune system based algorithm to train hidden Markov models, which is based on the principles of clonal selection, affinity maturation, vaccination and the abilities of learning and memory. Further, an integration algorithm based on the HMM and AIS for the MSA is constructed. Besides, a decoding algorithm based on Viterbi algorithm is also proposed to address the MSA problem. The proposed algorithm is validated by a set of standard instances taken from the benchmark alignment database. Numerical results show that the proposed algorithm is effective for multiple sequence alignment.

## Acknowledgements

The first three authors are grateful to the support of the national outstanding youth science fund (60625302), the National "973" Plan (2002CB3122000), the National "863" Plan (20060104Z1081), the Natural Science Fund of Shanghai (05ZR14038). The first author is grateful to the support of the China Post-Doctor Science Fund (20060400628)

## References

[1] L. Wang and T. Jiang, On the complexity of multiple sequence alignment, *Journal of Computational Biology*, 1: 337-348, 1994.

[2] Z.H. Du and F. Lin, Improvement of the Needleman-Wunsch algorithm, *Lecture Notes in Artificial Intelligence*, 3066: 792-797, 2004.

[3] F. Zhang, X.Z. Qiao and Z.Y. Liu, Parallel divide and conquer bio-sequence comparison based on Smith-Waterman algorithm, *Science in China Series F-Information Sciences*, 47 (2): 221-231, 2004.

[4] 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, *Nucl. Acids Res*, 22: 4673-4680, 1994.

[5] O. Goto, Optimal alignment between groups of sequences and its application to multiple sequence alignment, Comput. Appl. Biosci., 9: 361-370, 1993.

[6] A. Williams, D.R. Gilbert and D.R. Westhead, Multiple structural alignment for distantly related all beta structures using TOPS pattern discovery and simulated annealing, *Protein Engineering*, 16 (12): 913-923, 2003.

[7] C. Shyu and J.A. Foster, Evolving consensus sequence for multiple sequence alignment with a

genetic algorithm, *Lecture Notes in Computer Science*, 2724: 2313-2324, 2003.

[8] R.C. Edgar and K. Sjolander, Sequence alignment and tree construction using hidden Markov models, *Bioinformatics*, 19 (11): 1404-1411, 2003.

[9] J.R. Otterpohl, Baum-Welch learning in discrete hidden Markov models with linear factorial constraints, *Lecture Notes in Computer Science*, 2415: 1180-1185, 2002.

[10] P.S. Andrews and J. Timmis, Inspiration for the next generation of artificial immune systems, *Lecture Notes in Computer Science*, 3627, 126–138, 2005.

[11] X. Yue, A. Abraham, Z.X. Chi, Y.Y. Hao, and H.W. Mo, Artificial immune system inspired behavior-based anti-spam filter, *Soft Computing*, 11 (8): 729-740, 2007.

[12] S. Sahan, K. Polat, H. Kodaz, and S. Gunes, A new hybrid method based on fuzzy-artificial immune system and k-nn algorithm for breast cancer diagnosis, *Computers in Biology and Medicine*, 37 (3): 415-423, 2007.

[13] Z.H. Li and H.Z. Tan, A combinational clustering method based on artificial immune system and support vector machine, *Lecture Notes in Artificial Intelligence*, 4251:153-162, 2006.

[14] M. Chandrasekaran, P. Asokan, S. Kumanan, T. Balamurugan, and S. Nickolas, Solving job shop scheduling problems using artificial immune system, *International Journal of Advanced Manufacturing Technology*, 31(5-6): 580-593, 2006.

[15] J. Thompso, F. Plewniak and O. Poch, A comprehensive comparison of multiple sequence alignment programs, *Nucl. Acids. Res.*, 27 (13): 2682–2690, 1999.

[16] C.C. Chao, and Y.L. Yao, Hidden Markov models for the burst error statistics of Viterbi decoding, *IEEE Transactions on Communications*, 44(12): 1620-1622, 1996.