Available online at www.sciencedirect.com
ScienceDirect
Applied Soft Computing 8 (2008) 55–78
Applied Soft Computing
www.elsevier.com/locate/asoc

Genetic algorithm with ant colony optimization (GA-ACO) for multiple sequence alignment☆

Zne-Jung Lee [a,*], Shun-Feng Su [b], Chen-Chia Chuang [c], Kuan-Hung Liu [b]

[a] Department of Information Management, No. 1, Huafan Rd. Shihtin Hsiang, Taipei Hsien 223, Taiwan, ROC
[b] Department of Electrical Engineering, National Taiwan University of Science and Technology, Taiwan, ROC
[c] Department of Electrical Engineering, National Ilan University 1, Sec. 1, Shen-Lung Road, I-Lan 260, Taiwan, ROC

Received 26 September 2005; received in revised form 23 August 2006; accepted 29 October 2006
Available online 11 December 2006

Abstract

Multiple sequence alignment, known as NP-complete problem, is among the most important and challenging tasks in computational biology. For multiple sequence alignment, it is difficult to solve this type of problems directly and always results in exponential complexity. In this paper, we present a novel algorithm of genetic algorithm with ant colony optimization for multiple sequence alignment. The proposed GA-ACO algorithm is to enhance the performance of genetic algorithm (GA) by incorporating local search, ant colony optimization (ACO), for multiple sequence alignment. In the proposed GA-ACO algorithm, genetic algorithm is conducted to provide the diversity of alignments. Thereafter, ant colony optimization is performed to move out of local optima. From simulation results, it is shown that the proposed GA-ACO algorithm has superior performance when compared to other existing algorithms.

Keywords: Multiple sequence alignment; Genetic algorithm; Ant colony optimization; Hybrid search; Local search

1. Introduction

Multiple sequence alignment (MSA) is an essential tool for analyzing biological sequences. It is one of the most important and challenging tasks in computational biology because the time complexity for solving MSA grows exponentially with the size of the considered problem [1–3]. In general, heuristics offer more practical solutions but usually produce quasi-optimal alignment [4,5]. The progressive approach is the most commonly used heuristic method by gradually aligning the closest pair to build MSA [6–8]. Its main disadvantage is the found alignments that may by trapped in local optima, which stems from the greedy nature of this algorithm. This means that if mistakes are made in intermediate alignments, they cannot be corrected later when more sequences are added into the alignment process. Furthermore, there is no objective function to evaluate the performance for multiple sequence alignment. Another approach is to use an extension of dynamic programming for simultaneously aligning multiple sequences, such as Carrillo-Lipman algorithm [9], MSA [10], DCA [11,12], etc. In general, these algorithms often have higher quality solutions than that of progressive approach. However, the drawbacks of these algorithms are the complexity in running time and memory requirements. Thus, they can only be applied to problems with limited number of sequences [8].

There are also iterative, stochastic approaches for multiple sequence alignment. These approaches include simulated annealing (SA) [13], genetic algorithms (GAs) [14–18] and evolutionary programming (EP) [19–21]. SA is the first stochastic iterative method for simultaneously aligning a set of sequences, but its found alignments could be trapped in local optima [26]. Isokawa et al. applied a simple genetic algorithm with bit matrices to MSA [14]. Notredame et al. also used GAs to solve MSA and align two homologous RNA sequences [15,16]. However, GAs may cause certain degeneracy in search performance if their operators are not carefully designed [18,34,38,40]. Fogel proposed to use the evolutionary programming algorithm for multiple sequence alignment, and an extension version was proposed in the literature [21–23]. These algorithms work well for containing a large number of

☆ This work was supported by the National Science Council of Taiwan, R.O.C. under the grant NSC-93-2218-E-345-001, NSC 94-2213-E-211-013, and NSC 95-2221-E-211-023.
* Corresponding author.
E-mail address: johnlee@hfu.edu.tw (Z.-J. Lee).

1568-4946/$ – see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2006.10.012

fully matched blocks, but few fully matched blocks will be found for sequences with low similarity [20]. Another method based on the iteration idea is the Gibbs sampling [24]. Gibbs sampling has been successfully applied to the problem of multiple alignment block without gaps. These above algorithms provided the versatile and effective alignments for multiple sequence alignment. Nevertheless, they do not incorporate local search as an add-on extra to conduct fine tuning for the found alignments. Recently, genetic algorithms with local search have been considered as good alternatives for solving optimization problems [17,18,25–27]. The concept of local search is to find a better candidate nearby the current one before move to the next stage of search [26]. In the literature [28–34], the genetic algorithm equipped with local search is also referred to as the memetic algorithm, the genetic local search, the hybrid genetic algorithm, or the cultural algorithm. In this paper, we present the genetic algorithm with ant colony optimization (GA-ACO) to explore and exploit search spaces for multiple sequence alignment. It has both the advantage of GA, the ability to find feasible solutions and to avoid premature convergence, and that of ACO, the ability to search over the subspace and then to move out of local optima.

This paper is organized as follows. Because the proposed algorithm is based on genetic algorithm and ant colony optimization, we describe both approaches in Section 2. In Section 3, we present the proposed GA-ACO algorithm for multiple sequence alignment. Next, Section 4 shows the simulation results. The results are also compared to other existing algorithms to demonstrate the superiority of the proposed GA-ACO algorithm. Conclusions are given in Section 5.

2. Genetic algorithm and ant colony optimization

GA and ACO are population-based search algorithms by maintaining a population of structures as key elements in design and implementation of problem solving algorithms. These algorithms are sufficiently complex to provide powerful adaptive search approaches, and usually can be embedded with other approaches to speed up the search performance. The basics of GA and ACO are described later.

2.1. Genetic algorithm

GA has been touted as a class of general-purpose search strategies for optimization problems. GA use a population of solutions, from which, using recombination and selection strategies, better and better solutions can be produced. GA can handle any kind of objective functions and any kind of constraints without much mathematical requirements about the optimization problems. When applying to optimization problems, genetic algorithm provides the advantages to perform global search and hybridize with domain-dependent heuristics for specific problems [35]. Genetic algorithm starts with a set of randomly selected chromosomes as the initial population that encodes a set of possible solutions. In GA, variables of a problem are represented as genes in a chromosome, and the chromosomes are evaluated according to their fitness values using some measures of profit or utility that we want to optimize. Two genetic operators, crossover and mutation, alter the composition of genes to create new chromosomes called offspring. The selection operator is an artificial version of natural selection, a Darwinian survival of the fittest among populations, to create populations from generation to generation, and chromosomes with better fitness values have higher probabilities of being selected in the next generation. After several generations, GA can converge to the best solution.

2.2. Ant colony optimization

ACO algorithms have also widely used as a new cooperative search algorithm in optimization problems [36–38]. ACO algorithms are a class of algorithms inspired by the observation of real ants. Ants are capable of exploring and exploiting pheromone information, which have been left on the ground when they traversed. They then can choose routes based on the amount of pheromone. While building the solutions, each artificial ant collects pheromone information on the problem characteristics and uses this information to modify the representation of the problem, as seen by the other artificial ants [36]. The larger amount of pheromone is left on a route, the greater is the probability of selecting the route by artificial ants. In ACO, artificial ants find solutions starting from a start node and moving to feasible neighbor nodes in the process of ants_generation_and_activity. During the process, information collected by artificial ants is stored in the so-called pheromone trails. In the process, artificial ants can release pheromone while building the solution (online step-by-step) or while the solution is built (online delayed). An ant-decision rule, made up of the pheromone and heuristic information, governs artificial ants to search towards neighbor nodes stochastically. Pheromone_evaporation is a process of decreasing the intensities of pheromone trails over time. This process is used to avoid locally convergence and to explore more search space. Based on this property, we use ACO as local search in the proposed GA-ACO algorithm.

3. The proposed GA-ACO algorithm

The procedure of the proposed GA-ACO algorithm is to apply initialization, sexual reproduction operators, mutation operators, fitness evaluation, selection operator, and ACO for multiple sequence alignment. These above procedures are iterated until a pre-specified termination criterion is satisfied. The proposed algorithm is shown in Fig. 1. When the initial parent alignments are randomly generated, these alignment matrices are optimized by using the proposed algorithm. A population of parent alignments is maintained. Mutation and sexual reproduction operators are used to produce offspring alignments. The function of sexual reproduction operators is like the crossover operator in genetic algorithms. Sexual reproduction operators allow two alignments to exchange their information with each other and can produce one offspring alignment or two offspring alignments [20]. Mutation operators

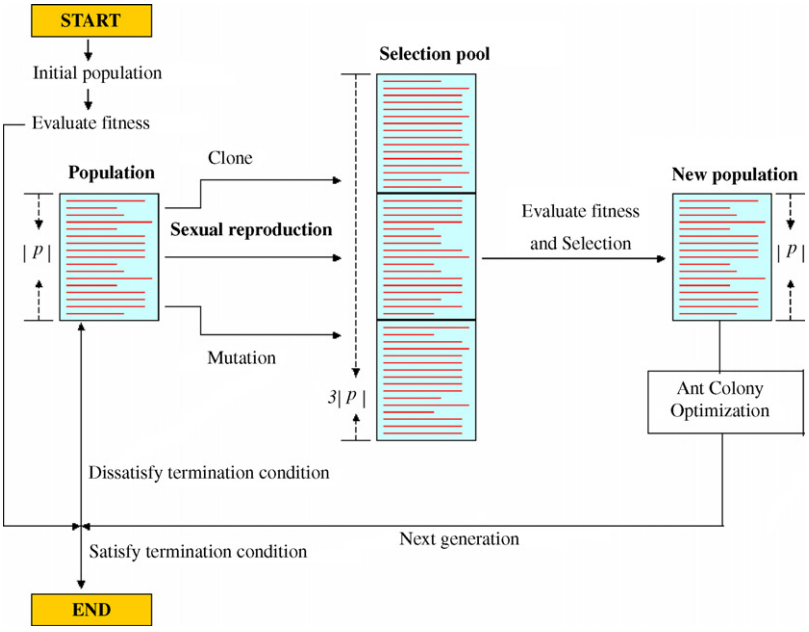

Fig. 1. The flowchart of the proposed algorithm.

provide possible variation on the alignments. After offspring alignments are generated via the mutation and sexual reproduction operators, the original parent alignments and all offspring alignments are put into a selection pool. The selection operator is employed to determine which alignments are suitable to survive and which alignments are culled in the next generation. Moreover, ACO is applied to the best alignment to escape from the local optima in each generation. In this section, we introduce all operators used in the proposed GA-ACO algorithm.

3.1. Representation

It is considered that there are k sequences to be aligned, and these sequences are generated with various lengths, say, from l_1 to l_k. In our approach, parent alignments are presented as a matrix where each sequence is encoded as a row with the considered alphabet set. The maximum number of columns in the matrix is $w = [\alpha\, l_{\max}]$, where $l_{\max} = \max\{l_1, l_2, \ldots, l_k\}$, $[x]$ is the smallest integer greater than or equal to x, and the parameter α a scaling factor [14,20]. In our study, each matrix may have variant number of columns. The value α is chosen according to the probability distribution $N(\mu, \sigma^2)$, where μ and σ^2 denote the mean and variance of Gaussian distribution.

3.2. Population initialization

The initial population P is generated by randomly initializing their constituent rows in the following ways. First, for each sequence s_i with length l_i, a random permutation is generated from the set $1, 2, \ldots, w$. Secondly, the first l_i numbers of the permutation are sorted in ascending order. Thirdly, those l_i numbers are used as the row indices in the matrix indicating where the corresponding base symbols taken from the original sequence s_i are placed. Finally, the positions for all rows in the matrix without associated base symbols are filled with the space symbols. Fig. 2 illustrates an example of

Sequence	Length	Permutation(1-9)	Positions	Sorted positions
ATCAA	(5)	352691748	35269	23569
TAATCAA	(7)	967148532	9671485	1456789
ATCA	(4)	625148379	6251	1256
TAATCAT	(7)	749135862	7491358	1345789
ATGATT	(6)	568431972	568431	134568

Initial Alignment

-	A	T	-	C	A	-	-	A
T	-	-	A	A	T	C	A	A
A	T	-	-	C	A	-	-	-
T	-	A	A	T	-	C	A	T
A	-	T	G	A	T	-	T	-

$$\alpha = N(1.3, 0.2), \text{in this case } \alpha = 1.2$$

$$w = \lceil \alpha \times l_{\max} \rceil = \lceil 1.2 \times 7 \rceil = 9$$

Fig. 2. Example illustrating the initialization procedure.

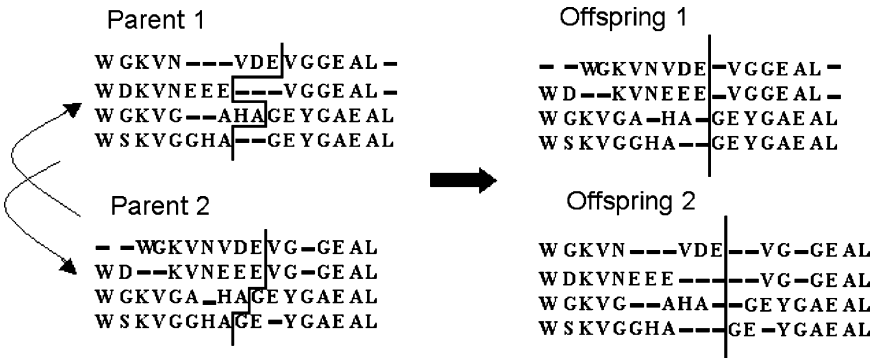

Fig. 3. Example illustrating the procedure of *singlepoint* operator.

using the random initialization procedure to generate the initial population.

3.3. Fitness evaluation

Each alignment represented as a matrix has a corresponding fitness value. To determine which alignment will survive in the next generation is based on its fitness value. In the proposed GA-ACO algorithm, the optimization task is defined as a maximization problem for searching the alignment with the best fitness value. The fitness value for two sequences m_i and m_j of alignment M is defined as follows:

$$\text{Fitness}\,(m_i, m_j) = \sum_{1 \le p \le N} C(m_{ip}, m_{jp}). \tag{1}$$

Then, the fitness value of an alignment M is defined as:

$$\text{Fitness}\,(M) = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \text{Fitness}(m_i, m_j). \tag{2}$$

where, N is the length of the alignment that must be greater than or equal to the length of the longest sequence, and C a scoring matrix.

3.4. Selection and terminations

In this paper, tournament selection is implemented as the selection operator. This selection scheme is to determine which alignments in the selection pool will become parent alignments in the next generation [25]. The termination condition is set as that the number of generation exceeds the permitted generation denoted as $g_{\max}$. When the termination condition is satisfied, the resultant alignment is obtained.

3.5. Sexual reproduction operators

The sexual reproduction operators include *singlepoint* operator, *recombinematchcolumn* operator [20], and *uniformexchangeblock* operator. The *singlepoint* operator produces two offspring alignments through exchanging information in parent alignments. The parent 1 alignment is cut at some randomly chosen positions, and parent 2 alignment is also tailored at the same ordered base position for each sequence. The left parts of both parent alignments are exchanged for keeping the original sequence base order, if there is a single cutting point in both parent alignments. To do above steps, extra spaces may be inserted into the junction points. Fig. 3 outlines this operator. The *recombinematchcolumns* operator tries to create an offspring alignment with match columns by swapping blocks and inserting spaces into the junction points [20]. Fig. 4 illustrates this operator. The *uniformexchangeblock* operator is to map the match columns that are consistent in parent alignments. Match columns are said to be consistent in parent alignments, if they contain the same ordered base. We randomly choose two consistent match columns as the two cutting points, and then blocks between two consistent match columns can be directly swapped [15]. This operator is outlined in Fig. 5.

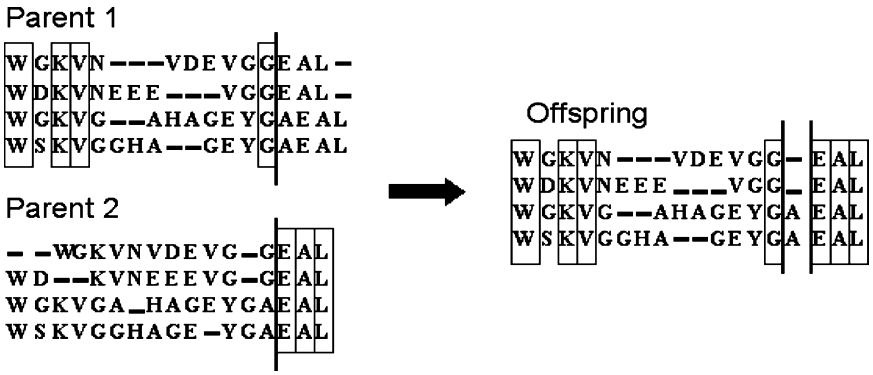

Fig. 4. Example illustrating the procedure of *recombinematchcolumn* operator.

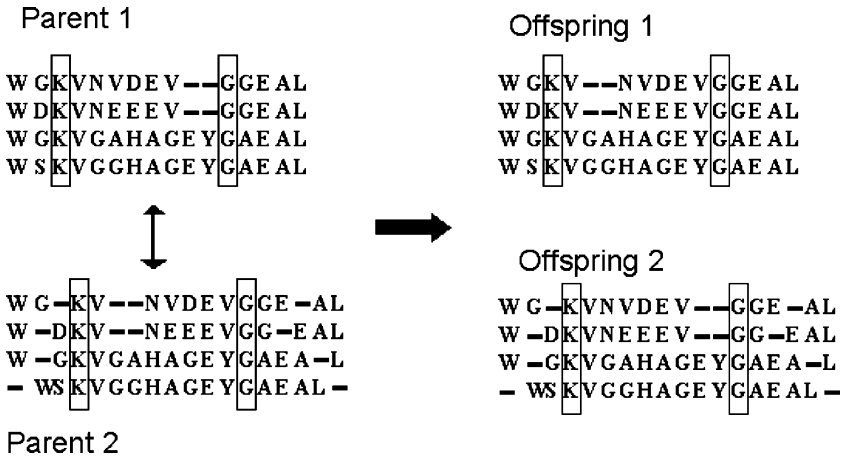

Fig. 5. Example illustrating the procedure of *uniformexchangeblock* operator.

3.6. Mutation operators

The mutation operators perform modification to provide possible variation for the offspring alignments. The mutation operators are classified into three categories: the space block shuffle operators, the base symbol block shuffle operators, and the heuristic operators. The difference between the space block shuffle operators and the base symbol block shuffle operators is that the block to be shuffled is different. The former operators shuffle the space blocks and the later operators shuffle the symbol base block. Both types of mutation operators are used in providing variety for offspring alignments. The heuristic operators perform modifications in a rather smart and subtle manner, and have the power of searching higher fitness values.

3.6.1. Space block shuffle operators

Space block shuffle operators include *changespace* operator and *mergespace* operator. The *changespace* operator randomly picks a space from a randomly chosen row in a parent alignment. Then, the randomly chosen space symbol is moved to a base symbol position selected randomly in the same row. There is an associated parameter with this operator denoted as λ, which is a self-adaptive parameter. It represents the number of rows to be selected for applying this operator, $\lambda = [0.2k]$ (k is the number of sequences). Fig. 6 illustrates an example of this operator. The *mergespace* operator merges two separated spaces together. Both spaces are randomly chosen in a parent alignment. This operator is given in Fig. 7. This operator is to move the block S to the right hand side and move the randomly

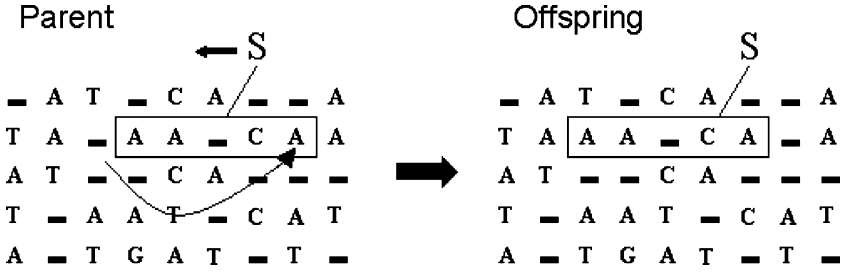

Fig. 6. Example illustrating the procedure of *changespace* operator.

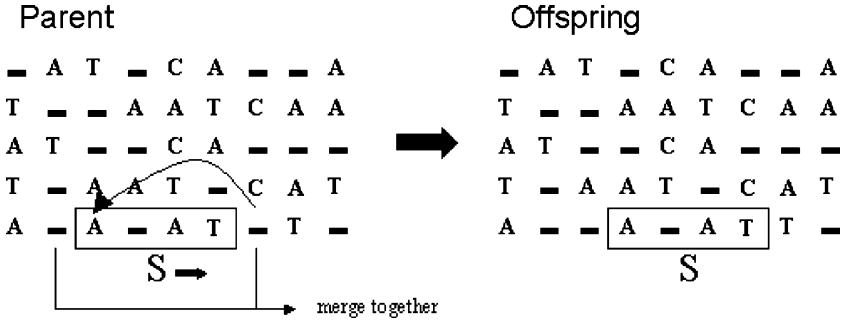

Fig. 7. Example illustrating the procedure *of mergespace* operator.

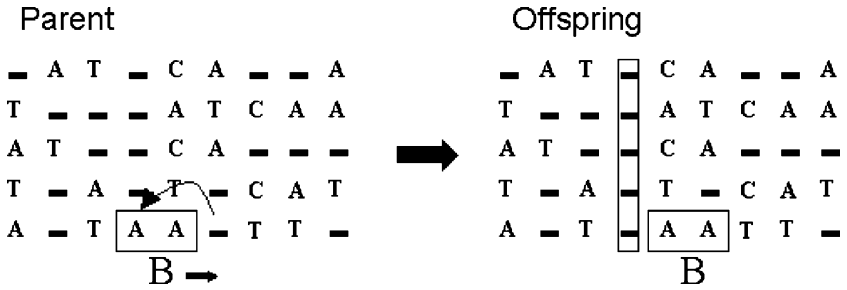

Fig. 8. Example illustrating the procedure of *decreasesearchspace* operator.

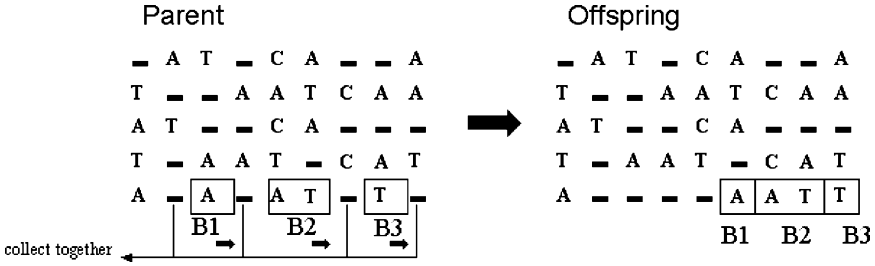

Fig. 9. Example illustrating the procedure of *collectspace* operator.

chosen space symbol to the destination position. In the *mergespace* operator, it also has the parameter λ representing the selected number of rows.

3.6.2. Base symbol block shuffle operators

Base symbol block shuffle operators include *decreasesearchspace* operator and *collectspace* operator. The *decreasesearchspace* operator decreases the alignment length. This operator chooses a space column with poor performance to make a full space column by grabbing the neighborhood spaces, and then it deletes the full space columns. This operator can delete the bad columns from the alignment and distribute the base symbols to another column. It is shown in Fig. 8. In Fig. 8, the ending symbol 'A' grabs a space symbol if the space column 4 is chosen. The space symbol is grabbed in the end of column 6 and then the base symbol block *B* is shuffled to the right hand side. The *collectspace* operator collects separated spaces together. The *collectspace* operator is given in Fig. 9. First, one space is randomly chosen from a parent alignment. Then, the separated spaces closed to the chosen space are collected together. This operation is to shuffle the base symbol blocks *B*1, *B*2, *B*3 to the right hand side, hereafter the closed

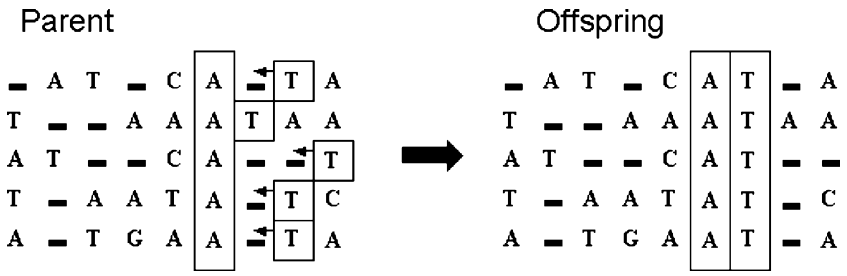

Fig. 10. Example illustrating the procedure of *catchmatchcolumn* operator.

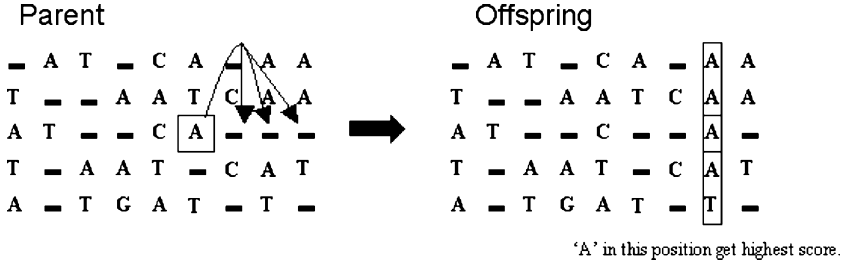

Fig. 11. Example illustrating the procedure of *movetohigherscoreposition* operator.

Table 1
List of data sets

ID	Number of sequences	Mean sequences length (min, max)	Similarity
D1	10	212 (211, 212)	0.9340
D2	8	457 (457, 457)	0.9825
D3	21	122 (122, 122)	0.8934
D4	21	334 (324, 348)	0.2969
PD1	4	199 (186, 208)	0.2931
PD2	6	493 (491, 498)	0.5691
PD3	10	348 (337, 352)	0.5057
PD4	15	151 (145, 163)	0.1726

three space symbols are collected together. In this operator, it also has a parameter λ representing the selected number of rows.

3.6.3. Heuristic operators

The Heuristic operators include *catchmatchcolumn* operator and *movetohigherscoreposition* operator. The *catchmatchcolumn* operator randomly selects a match column in a parent alignment. If possible, this operator tries to swap base symbol to space symbol, hereafter a new adjacent match column is generated either to the left or to the right of this selected match column [21]. Fig. 10 shows an example of this operator. In Fig. 10, the 6-th column is a match column with the base symbol 'A', and it is selected to generate adjacent match columns. For the 7-th column, the match column of 'T' can be found to generate a new match column. The *movetohigherscoreposition* operator randomly selects one row in a parent alignment, and then it scans through this row to find all base symbols that have one or more adjacent space symbols. Notice that the selected base symbol can be shifted to the left or right (or both) due to the presence of adjacent space symbols. The fitness value of this alignment is recomputed if the selected base symbol is shifted to neighboring space symbols. The position with the highest fitness value is selected for the selected base symbol. Fig. 11 shows this operator. The *movetohigherscoreposition* operator also has a parameter λ representing the selected number of rows.

3.7. Ant colony optimization as local search

In this paper, we use ACO embedded into GA as local search to enhance the search performance. It is noted that a sliding-window (N) is used to detect mismatch blocks [8], and then ACO is used to ameliorate these poorly aligned regions. When the best alignment has been found in each generation, the pheromone trail update rule for sequence m_i ($i = 1, \ldots, k-1$) and m_j ($j = i+1, \ldots, k$) is performed as:

$$\tau_{ijl}(t+1) = (1-\rho)\tau_{ijl}(t) + \rho\Delta\tau_{ij}(t) \qquad 1 \le l \le N \tag{3}$$

where $0 < \rho \le 1$ is a parameter governing the pheromone decay process, l the position of residue, $\Delta\tau_{ij}(t) = \sum_{i=1}^{k-1}\sum_{j=i+1}^{k} W_{ij} \times \mathrm{Score}(A_{ij}) / \sum_{i=1}^{k-1}\sum_{j=i+1}^{k} W_{ij} \times \mathrm{Len}$ then W_{ij} is the percent identity between two aligned sequences m_i and m_j, A_{ij} is the pairwise projection of sequence m_i and m_j from the multiple sequence alignment, Score(A_{ij}) is the number of aligned pairs of residues that are shared between A_{ij} and the library, and Len is the length of this alignment [39]. The global best alignment is defined as the alignment with best fitness value found from the beginning of generation and set as pairwise library [39]. In the ant's generation and activities of ACO, two sequences m_i and m_j with mismatch blocks are randomly selected. The ant-decision table for the ant s selecting the l-th residue in the mismatch blocks is governed by:

$$\pi(l) = \begin{cases} \arg\left\{ \max_{l=\mathrm{allowed}_s(t)} \left[\tau_{ijl}(t)\, \eta_{ijl}^{\beta} \right] \right\} & \text{when } (q \le q_0) \\ L & \text{otherwise} \end{cases} \tag{4}$$

Table 2
The comparison of the proposed algorithm, ClustalW, genetic algorithm without ACO, central-star algorithm, and Horng's genetic algorithm

ID	The proposed GA-ACO algorithm	ClustalW	Genetic algorithm without ACO as local search	Central-star	Horng's genetic algorithm [40]
D1	18627	18627	18627	18627	18627
D2	25343	25343	25343	25343	25343
D3	47889	47889	47889	47889	47889
D4	84394	81769	81769	81760	81769
PD1	12585	12418	12481	12273	12418
PD2	89775	89634	89693	89388	89634
PD3	193823	193632	193625	192969	193632
PD4	166982	161654	166135	160059	166573

where η_{ijl} is the heuristic information and is set as $\eta_{ijl} = -\max[C(m_{il}, m_{jl})1]$, β a parameter representing the importance of heuristic information, q a random number uniformly distributed in [0, 1], q_0 a pre-specified parameter ($0 \leq q_0 \leq 0.1$), $\text{allowed}_s(t)$ is the set of ordered and feasible residues for the ant s at time t. The probability distribution is given by:

$$P(t) = \begin{cases} \dfrac{\tau_{ijl}(t)\eta_{ijl}^{\beta}}{\sum_{u \in \text{allowed}_s(t)} \tau_{iju}(t)\eta_{iju}^{\beta}} & \text{if } l \in \text{allowed}_s(t) \\ 0, & \text{otherwise;} \end{cases} \quad (5)$$

In finding feasible solutions, ants perform online step-by-step pheromone update rule as:

$$\tau_{ijl}(t+1) \leftarrow (1-\psi)\tau_{ijl}(t) + \psi\Delta\tau_s \quad (6)$$

where $0 < \psi \leq 1$ is a constant, and $\Delta\tau_s = 1/k|C(m_{il}, m_{jl})|$ is found by the ant s.

4. Simulation and results

In simulation, we need to identify a set of parameters. We performed the simulation with various values, and the results are all similar. Experiments were conducted on PCs with P4 3 GHz processor. In the following simulations, we keep the following values as default: $\rho = 0.1$, $\psi = 0.1$, $q_0 = 0.8$, $\beta = 2$, number of ant = 20, population size = 50, and $g_{\max} = 5000$. For genetic operators, the sexual reproduction probability (P_c) is set as 0.5 and the mutation probability (P_m) is 0.05. Thereafter, they are equi-probable for selecting corresponding operators in sexual reproduction operators and mutation operators. In simulation, the data sets used in the experiments are SWISS-PROT benchmark alignment database [22]. The data sets D1, D2, D3, and D4 are DNA or RNA types. The data sets from D1 to D3 are very similar but D4 data set has a poor similarity measure. The data sets PD1, PD2, PD3, and PD4 are protein types. The information for all data sets is shown in Table 1. For comparison, the fitness value is used to evaluate the search performance. The match, mismatch, and space scores are +2, −1, and −2, respectively, for DNA and RNA sequences. The scoring matrix and the space score are PAM250 and −1, respectively, for protein sequences [22]. In this experiment, each test data is running for 10 -imes to average the results. These alignments were also constructed by using ClustalW [7], Gusfield's central-star algorithm [35], GA without ACO as local search, and the proposed GA-ACO algorithm. We also compare the simulation results with a genetic algorithm proposed by Horng et al. [40]. It is noted that the algorithm of GA without ACO uses the same genetic operators and the same CPU time requirements as in the proposed GA-ACO algorithm. The best results obtained by the proposed GA-ACO algorithm are shown from Figs. 12–19. Table 2 tabulates all simulation results. When the similarity of data set is high, all algorithms almost get the same performance. When the similarity of data set is low, the proposed GA-ACO algorithm has much better performance than other algorithms. It is clearly evident that the proposed GA-ACO algorithm indeed has the best results for all data sets.

5. Conclusions

Most genetic algorithms have focused on genetic operators, crossover and mutation operators, for multiple sequence alignment. In this paper, we present a novel algorithm of genetic algorithm with ant colony optimization for multiple sequence alignment. In the proposed GA-ACO algorithm, ACO as local search is embedded into GA to escape from local optima and ameliorate the search performance. From simulation results, it is shown that the proposed algorithm can improve search performance and outperform other existing algorithms.

Acknowledgements

The authors would like to thank the anonymous reviewers for their comments and constructive suggestions that have improved the paper. This work was supported by the National Science Council of Taiwan, R.O.C. under the grant NSC-93-2218-E-345-001, NSC 94-2213-E-211-013, and NSC 95-2221-E-211-023.

Appendix

Figs. 12–19.

```
HCV2L1A10    gtgctgcctggtacatcaagggcaggctggtccctggggcggcgtacgctttctacggcg
HCV2L3A5     gtgctgcctggtacatcaagggcaggctggtccctggggcggcgtatgctttctatggcg
HCV2L3A7     gtgctgcctggtacatcaagggcaggctggtccctggggcggcgtacgctttctacggcg
HCV2L3A9     gtgctgcctggtacatcaagggcaggctggtccctggggcggcgtacgctttctacggcg
HCV2L3B1     gtgctgcctggtacatcaagggcaggctggtccctggggcggcgtacgctttctacggcg
HCV2L3B2     gtgctgcctggtacatcaagggcaggctggtccctggggcggcgtacgctttctacggcg
HCV2L3C1     gtgctgcctggtacatcaagggcaggctggtccctggggcggcgtacgctttctacggcg
HCV2L3C8     gtgctgcctggtacatcaagggcaggctggtccctggggcggcgtacgccttctacggcg
HCV2L3D4     gtgctgcctggtacatcaagggcaggctggtccctggggcggcgtacgctttctacggcg
HCV2L3E6     gtgctgcctggtacatcaagggcaggctggtccctggggcggcgtacgctttctacggcg
             ********************************************.**.*****.****

HCV2L1A10    tatggccgctgctcctgctcctgttggcgttgccaccacgtgcatacgccatggaccggg
HCV2L3A5     tatggccgctgctcctgctcctgttggcgttgccaccacgtgcatacgccatggaccggg
HCV2L3A7     tatggccgctgctcctgctcctgttggcgttgccaccacgtgcatacgccatggaccggg
HCV2L3A9     tatggccgctgctcctgctcctgttggcgttgccaccacgtgcatacgccatggaccggg
HCV2L3B1     tatggccgctgctcctgctcctgttggcgttgccaccacgtgcatacgccatggaccggg
HCV2L3B2     tatggccgctgctcctgctcctgttggcgttgccaccacgtgcatacgccatggaccggg
HCV2L3C1     tatggccgctgctcctgctcctgttggcgttgccaccacgtgcatacgccatggaccggg
HCV2L3C8     tatggccgctgctcctgctcctgttggcgttgccaccacgtgcatacgccatggaccggg
HCV2L3D4     tatggccgctgctcctgctcctgttggcgttgccaccacgtgcgtacgccatggaccggg
HCV2L3E6     tatggccgctgctcctgctcctgttggcgttgccaccacgtgcatacgccatggaccggg
             ****************************************.****************

HCV2L1A10    agatggctgcatcgtgcggaggcgcggtccttgtgggtctgatactcttgaccttgtcac
HCV2L3A5     agatggc-gcatcgtgcggaggcgcggtccttgtgggtctgatactcttgaccttgtcgc
HCV2L3A7     agatggctgcatcgtgcggaggcgcggtccttgtgggtctgatactcttgaccttgtcac
HCV2L3A9     agatggctgcatcgtgcggaggcgcggtccttgtgggtctgatactcttgaccttgtcac
HCV2L3B1     agatggctgcatcgtgcggaggcgcggtccttgtgggtctgatactcttgaccttgtcac
HCV2L3B2     agatggctgcatcgtgcggaggcgcggtccttgtgggtctgatactcttgaccttgtcac
HCV2L3C1     agatggctgcatcgtgcggaggcgcggtccttgtgggtctgatactcttgaccttgtcac
HCV2L3C8     agatggctgcatcgtgcggaggcgcggtccttgtgggtttgatactcttgaccttgtcac
HCV2L3D4     aggtggctgcatcgtgcggaggcgcggtccttgtgggtctgatactcttgaccttgttac
HCV2L3E6     agatggctgcatcgtgcggaggcgcggtccttatcggtctgatactcttgaccttgtctc
             **.****.************************.*.***.******************..*

HCV2L1A10    cacactataaagtgttcctcgctaagctcata
HCV2L3A5     cacactataaagtgttcctcgctaagctcata
HCV2L3A7     cacactataaagtgttcctcgctaagctcata
HCV2L3A9     cacactataaagtgttcctcgctaaactcata
HCV2L3B1     cacactataaagtgttcctcgctaagctcata
HCV2L3B2     cacactataaagtgttcctcgctaaactcata
HCV2L3C1     cacactataaagtgttcctcgctaagctcata
HCV2L3C8     cgcactataaagtgttcctcgctaagctcata
HCV2L3D4     cgcactataaagtgttcctcgccaagctcata
HCV2L3E6     cacactataaagtgttcctcgctaagctcata
             *.********************.**.******
```

Fig. 12. Alignment result of data set D1 by the proposed algorithm. An asterisk is used to indicate a match column.

```
AB023276    acttcaggacggggacaacagttggaaacgactgctaatacccgatgtgccgcaaggtga
AB023278    acttcaggacggggacaacagttggaaacgactgctaatacccgatgtgccgcgaggtga
AB023279    acttcaggacggggacaacagttggaaacgactgctaatacccgatgtgccgcaaggtga
AB023283    acttcaggacggggacaacagttggaaacgactgctaatacccgatgtgccgaaaggtga
B023284     acttcaggacggggacaacagttggaaacgactgctaatacccgatgtgccgaaaggtga
AB023285    acttcaggacggggacaacagttggaaacgactgctaatacccgatgtgccgcaaggtga
AB023286    acttcaggacggggacaacagttggaaacgactgctaatacccgatgtgccgcaaggtga
AB023287    acttcaggacggggacaacagttggaaacgactgctaatacccgatgtgccgcaaggtga
            **************************************************..******

AB023276    aacctaattggcctggagaagagcttgcgtctgattagctagttggtggggtaaaggcct
AB023278    aacctaattggcctgaagaagagcttgcgtctgattagctagttggtggggtaaaggcct
AB023279    aacctaattggcctggagaagagcttgcgtctgattagctagttggtggggtaaaggcct
AB023283    aacctaattggcctgaagaagagcttgcgtctgattagctagttggtggggtaagagcct
B023284     aacctaattggcctgaagaagagcttgcgtctgattagctagttggtggggtaaaggcct
AB023285    aacctaattggcctgaagaagagcttgcgtctgattagctagttggtagggtaaaagcct
AB023286    aacctaattggcctgaagaagagcttgcgtctgattagctagttggtagggtaaaagcct
AB023287    aacctaattggcctgaagaagagcttgcgtctgattagctagttggtagggtaaaagcct
            **************.******************************.******..****

AB023276    accaaggcgacgatcagtagctggtctgagaggatgagcagccacactgggactgagaca
AB023278    accaaggcgacgatcagtagctggtctgagaggatgagcagccacactgggactgagaca
AB023279    accaaggcgacgatcagtagctggtctgagaggatgagcagccacactgggactgagaca
AB023283    accaaggcgacgatcagtagctggtctgagaggatgagcagccacactgggactgagaca
B023284     accaaggcgacgatcagtagctggtctgagaggatgagcagccacactgggactgagaca
AB023285    accaaggcgacgatcagtagctggtctgagaggatgagcagccacactgggactgagaca
```

Fig. 13. Alignment result of data set D2 by the proposed algorithm.

```
AB023286    accaaggcgacgatcagtagctggtctgagaggatgagcagccacactgggactgagaca
AB023287    accaaggcgacgatcagtagctggtctgagaggatgagcagccacactgggactgagaca
            ************************************************************

AB023276    cggcccagactcctacgggaggcagcagtggggaattttccgcaatgggcgaaagcctga
AB023278    cggcccagactcctacgggaggcagcagtggggaattttccgcaatgggcgaaagcctga
AB023279    cggcccagactcctacgggaggcagcagtggggaattttccgcaatgggcgaaagcctga
AB023283    cggcccagactcctacgggaggcagcagtggggaattttccgcaatgggcgaaagcctga
B023284     cggcccagactcctacgggaggcagcagtggggaattttccgcaatgggcgaaagcctga
AB023285    cggcccagactcctacgggaggcagcagtggggaattttccgcaatgggcgaaagcctga
AB023286    cggcccagactcctacgggaggcagcagtggggaattttccgcaatgggcgaaagcctga
AB023287    cggcccagactcctacgggaggcagcagtggggaattttccgcaatgggcgaaagcctga
            ************************************************************

AB023276    cggagcaacgccgcgtgagggaggaaggtctttggattgtaaacctcttttctcaaggaa
AB023278    cggagcaacgccgcgtgagggaggaaggtctttggattgtaaacctcttttctcaaggaa
AB023279    cggagcaacgccgcgtgagggaggaaggtctttggattgtaaacctcttttctcaaggaa
AB023283    cggagcaacgccgcgtgagggaggaaggtctttggattgtaaacctcttttctcaaggaa
B023284     cggagcaacgccgcgtgagggaggaaggtctttggattgtaaacctcttttctcaaggaa
AB023285    cggagcaacgccgcgtgagggaggaaggtctttggattgtaaacctcttttctcaaggaa
AB023286    cggagcaacgccgcgtgagggaggaaggtctttggattgtaaacctcttttctcaaggaa
AB023287    cggagcaacgccgcgtgagggaggaaggtctttggattgtaaacctcttttctcaaggaa
            ************************************************************

AB023276    gaagttctgacggtacttgaggaatcagcctcggctaactccgtgccagcagccgcggta
AB023278    gaagttctgacggtacttgaggaatcagcctcggctaactccgtgccagcagccgcggta
AB023279    gaagttctgacggtacttgaggaatcagcctcggctaactccgtgccagcagccgcggta
AB023283    gaagttctgacggtacttgaggaatcagcctcggctaactccgtgccagcagccgcggta
B023284     gaagttctgacggtacttgaggaatcagcctcggctaactccgtgccagcagccgcggta
AB023285    gaagttctgacggtacttgaggaatcagcctcggctaactccgtgccagcagccgcggta
AB023286    gaagttctgacggtacttgaggaatcagcctcggctaactccgtgccagcagccgcggta
AB023287    gaagttctgacggtacttgaggaatcagcctcggctaactccgtgccagcagccgcggta
            ************************************************************

AB023276    atacgggggaggcaagtgttatccggaattattgggcgtaaagcgtccgcaggtggtcag
AB023278    atacgggggaggcaagcgttatccggaattattgggcgtaaagcgtccgcaggtggtcag
AB023279    atacgggggaggcaagcgttatccggaattattgggcgtaaagcgtccgcaggtggtcag
AB023283    atacgggggaggcaagcgttatccggaattattgggcgtaaagcgtccgcaggtggtcag
```

Fig. 13. (*Continued*)

```
B023284     atacgggggaggcaagcgttatccggaattattgggcgtaaagcgtccgcaggtggtcag
AB023285    atacgggggaggcaagcgttatccggaattattgggcgtaaagcgtccgcaggtggtcag
AB023286    atacgggggaggcaagcgttatccggaattattgggcgtaaagcgtccgcaggtggtcag
AB023287    atacgggggaggcaagcgttatccggaattattgggcgtaaagcgtccgcaggtggtcag
            ***************.****************************************

AB023276    ccaagtctgccgtcaaatcaggttgcttaacgaccta
AB023278    ccaagtctgctgtcaaatcaggttgcttaacgaccta
AB023279    ccaagtctgccgtcaaatcaggttgcttaacgaccta
AB023283    ccaagtctgccgtcaaatcaggttgcttaacgaccta
B023284     ccaagtctgccgtcaaatcaggttgcttaacgaccta
AB023285    ccaagtctgccgtcaaatcaggttgcttaacgaccta
AB023286    ccaagtctgccgtcaaatcaggttgcttaacgaccta
AB023287    ccaagtctgccgtcaaatcaggttgcttaacgaccta
            **********.**************************
```

Fig. 13. (*Continued*).

```
1   aatctgtggggcttcttaataccaccggtggcgggggcagactttctagcagccttggaa
2   aatctgtggggtttcttaataccaccggtggcgggggcagactttcttgcagccttggaa
3   aatctgtggggcttcttaataccaccggtagcgggggcggactttctggcggccttggaa
4   aatctgtggggtttcttaataccaccggtggcgggggcagactttctagcagccttggaa
5   aatctgtggggcttcttaataccaccggtagcgggggcggactttctggcggccttggaa
6   aatctgtggggcttcttaataccaccggtggcgggggcggactttctggcggccttggaa
7   aatctgtggggcttcttgataccaccggtagcgggggcagactttctagcggccttggaa
8   aatctgtggggcttcttaataccaccggtggcgggggcggactttctggcggccttggaa
9   aatctgtggggtttcttaataccaccggtggcgggggcagactttctagcagccttggaa
10  aatctgtggggcttcttaataccaccggtggcgggggcggactttctggcggccttggaa
11  aatctgtggggcttcttgataccaccagtggcgggggcagactttctggcggccttggaa
12  aatctgtggggcttcttaataccaccggtggcgggggcggactttctggcggccttggaa
13  aatctgtggggtttcttaataccaccggtggcgggggcagactttctagcagccttggaa
14  aatctgtggggtttcttaataccaccggtggcgggggcagactttctagcagccttggaa
15  aatctgtggggcttcttaataccaccggtggcgggggcagactttctagcagccttggaa
16  aatctgtggggcttcttaataccaccggtggcgggggcagactttctggcggccttggaa
17  aatctgtggggtttcttaataccaccggtggcgggggcagactttctagcagccttggaa
18  aatctgtggggcttcttaataccaccggtggcgggggcggactttctggcggccttggaa
19  aatctgtggggcttcttaataccaccggtagcgggggcggactttctggcggccttggaa
20  aatctgtggggtttcttaataccaccggtggcgggggcagactttctagcagccttggaa
21  aatctgtggggcttcttaataccaccggtggcgggggcggactttctggcggccttggaa
    ***********.*****.********.**.********.********.**.*********

1   gcgagttgctttctgggggccttagcaccagttgattttctagcggtttatttagttcta
2   gcgagttgctttctgggggccttagcaccagttgattttctagcggtttatttagttcta
3   gcgagttgctttctgggagccttagcaccagtagactttctagcagtttatttagttcta
4   gcgagttgctttctgggggccttagcaccagttgattttctagcggtttatttagttcta
5   gcgagttgctttctgggagccttagcaccagtagactttctagcagtttatttagttcta
6   gcgagttgctttctgggagccttagcaccagtagactttctagcggtttatttagttcta
7   gcgagttatttttctgggggccttagcaccagtggactttctagcagtttatttagttcta
8   gcgagttgctttctgggagccttagcaccagtagactttctagcggtttatttagttcta
9   gcgagttgctttctgggggccttagcaccagttgattttctagcggtttatttagttctagc
10  gcgagttgctttctgggagccttagcaccagtagactttctagcggtttatttagttctagc
11  gcgagttatttttctgggggccttagcaccagtggactttctagcagtttatttagttctagc
12  gcgagttgctttctgggagccttagcaccagtagactttctagcagtttatttagttctagc
13  gcgagttgctttctgggggccttagcaccagttgattttctagcggtttatttagttctagc
14  gcgagttgctttctgggggccttagcaccagttgattttctagcggtttatttagttctagc
15  gcgagttgctttctgggggccttagcaccagttgattttctagcggtttatttagttctagc
16  gcgagttatttttctgggagccttagcaccagtagactttctagcggtttatttagttctagc
17  gcgagttgctttctgggggccttagcaccagttgattttctagcggtttatttagttctagc
18  gcgagttactttctgggagccttagcaccagtagactttctagcggtttatttagttctagc
19  gcgagttgctttctgggagccttagcaccagtagactttctagcagtttatttagttctagc
20  gcgagttgctttctgggggccttagcaccagttgattttctagcggtttatttagttctagc
21  gcgagttgctttctgggagccttagcaccagtagactttctagcggtttatttagttctagc
    *******..********.**************.**.********.*****************
```

Fig. 14. Alignment result of data set D3 by the proposed GA-ACO algorithm.

```
1   catttct---tt---tt-a-gggat-t--t-ta-a-aagttgtc--tt-tt--ct--t--
2   catttct---tt---tt-aaggg-t-t--t-ta-a-aaattgtc--tt-tt---t--t--
3   catttct---tt---tt-aaggg-t-t--t-ta-a-aaattgtc--tt-tt--ct--t--
4   cattt-t---ttc--tt-aagtg-t-t--t-tg-g-tatttatct-tt-tt--ct--t--
5   cattt-t---tgc--tt-atgta-t-t--tatagt-gggttgtct-tt-ttgact--t--
6   catttct---tt---tg-aagtg-a-t--t-tg-a-gatttatct-tt-tt--ct--t--
7   catttct---tt---tt-aaggg-t-t--t-ta-a-aaattgtc--tt-tt--ct--t--
8   catttct---tt---tt-atg---t-t--g-ag-a-tatttgtctgtt-tt--ct--t--
9   cattt-t---ta---ct-atgtg-t-tgat-tg-t-ggattgtct-tt-tt--ct--t--
10  catttctt--tta--tt-gagtg-a-a--g-aagagattttgtc--ttgtt--ttgat--
11  cattt-t---tc---tt-a-gtg-t-t--t-tg-g-tatttatct-tt-tt--ct--t--
12  catttct---tt---t--aaggg-t-t--t-ta-a-aaattgtc--tt-tt--ct--t--
```

Fig. 15. Alignment result of data set D4 by the proposed GA-ACO algorithm.

```
13    cattt-t---ttc--tt-a-gtg-t-t--t-tg-g-tatttatct-tt-tt--ct--t--
14    catttct---tt---tg-a-gggat-t--t-ta-a-aagttgtc--tt-tt---t--t--
15    catttct---tt---tg-a-gggat-t--t-ta-a-aagttgtc--tt-tt--ct--t--
16    catttct---tt---tt-atg---t-t--g-ag-a-tatttgtctttt-tt--ct--t--
17    catttct---tt---tt-aaggg-t-t--t-ta-a-aaattgtc--tt-tt--ct--t--
18    cattt-t---tc---tt-a-gtg-t-t--t-tg-g-tatttatct-tt-tt--ct--t--
19    catttct---ttggatg-aagag-a-t--t-tg-g-g-tttgtt--tt-tt--ct--t--
20    catttct---tt---t--aaggg-t-t--t-ta-a-aaattatc--tt-tt--ct--t--
21    cattttagattg--ttgaagag-tagg-t-tg-a----ttgt---tt-tt--at--taa
      *****.*...*.........*..................**.*...**.**...*..*..

1     --tt-c-t-t-g--tg-tatt-g-attt-tt-tt-t-taataaaattctttatgagctag
2     --ct-t-----g--tg-tatt-g-attt-tt-tt-t-taataaaattctttatgagctat
3     --tt-tat-t-g--tg-tatt-g-attt-tt-tt-t-taataaaattctttatgagctat
4     --tt-gcc-g-g--ta--att---attt-tt-tt-t-taataaaattctttatgaacaac
5     --ttcttt-t-gaaggttatt---attt-tt-tt-t-taataaaattctttatcgacaac
6     --tt-gag---ga-t--tat--g-attt-tt-ttgt-taataaaattctttatggccaac
7     --tt-tat-t-g--tg-tatt-g-attt-tt-tt-t-taataaaattctttatgagctat
8     --tt-gaa-g-g--t--tatt---attt-tt-tt-t-taataaaattctttatggacaac
9     --tt-gaa-g-g--t--tatt---attt-tt-tt-t-taataaaattctttat-agaaaa
10    --tt-tttct-t--t--tgtt---attt-tt-tt-tgtaataaaattctttatgctcaaa
11    --tt-gaa-g-g--t--tatt---attt-tt-tt-t-ttataaaatttttttatgaagaac
12    --tt-tat-t-g--tg-tatt-g-attt-tt-tt-t-taataaaattctttatgagctat
13    --tt-gaa-g-g--t--tatt---attt-tt-tt-t-taataaaatttttttatgaagaac
14    --ct-t-----g--tg-tatt-g-attt-tt-tt-t-taataaaattctttatgagctat
15    --tt-c-t-t-g--tg-tatt-g-attt-tt-tt-t-taataaaattctttatgagcttt
16    --tt-gaa-g-g--t--tatt---attt-tt-tt-t-taataaaattctttatggacaac
17    --tt-tat-t-g--tg-tatt-g-attt-tt-tt-t-taataaaattctttatgagctat
18    --tt-gaa-g-g--t--tatt---attt-tt-tt-t-taataaaatttttttatgaagaac
19    --ttgta----ga-t--tat--g-atttgtt-tg-t-taataaaattctttatggccaac
20    --tt-tat-t-g--tg-tatt-g-attt-tt-tt-t-taataaaatcctttatgagctat
21    agtt-ttt-tggact--tttaggaatttattatt-t-taatatttttttatgaaggggatg
      ...*..............*....****.**.*..*.*.***...*...*.*........

1     caatcatgcga-tctcatttgaaat-gtttagaattatcc-tgcgat-t-cgaagatatg
2     caatcatgcga-tctcatttgaaat-gtttggaattatcc-tgtgat-t-cgaagatatg
3     caatcatgcga-tctcatttgaaat-gtttggaattatcc-tgtgat-t-cgaagatatg
4     agttgaggata-gatcttttgaaat-gtttggagttatcc-ag-gat-tgagaagata--
```

Fig. 15. (*Continued*)

```
5   aattagggcaa-gatcatttgaaat-gtttggcttaatcc-tg-gatagac-aagatatg
6   agcggaggcga-gatcatttgaaat-gtttggaattatccaggagag---cgaagatatg
7   caatcatgcga-tctcatttgaaat-gtttggaattatcc-tgtgat-t-cgaagatatg
8   tttagagggta-gatcatttgaaat-gtttggaattatcc-tat-ac-g-cgc-gat-t-
9   caattagggtaagatcatttgaaat-gtttggcataatcc-ta-gatag-agaagatatg
10  aatcgggggta-gatcatttgaaat-gtttggagttatccaagggag-a--gaagatatt
11  tgtcaaggtta-gatcttttgaaat-gtttggaattatcc-ag-gat-tgagaagatacc
12  caatcatgcga-tctcatttgaaat-gtttggaattatcc-tgtgat-t-cgaagatatg
13  tgtcaaggtta-gatcttttaaaat-gtttggaattatcc-ag-gat-tgagaagatacc
14  caatcatgcga-tctcatttgaaat-gtttggaattatcc-tgcgat-t-cgaagatatg
15  caatcatacga-tctcatttgaaat-gtttggaattatcc-tgcgat-t-cgaagatata
16  tttagagggta-gatcatatgaaat-gtttggaattatcc-tat-ac-g-cgc-gat-t-
17  caatcatgcga-tctcatttgaaat-gtttggaattatcc-tgtgat-t-cgaagatatg
18  tgtcaaggtta-gatcttttgaaat-gtttggaattatcc-ag-gat-tgagaagatacc
19  aaaaaaggcga-gatcatttgaaat-gtttggaattatccaggagag---cgaagatatg
20  caatcatgcga-tctcatttgaaat-gtttggaattatcc-tgtgat-t-cgaagatatg
21  gagtc-ggc-a-g-tgttttgaattagtgtg-aattatcc-aa-gatct-taattaaagg
    ..........*...*..*.*.**.*.**.*....*.****.....*.........*....

1   cctt--tttttgattggatactctctaaggaaatcagattttaaagaaactatccaatca
2   ccct--tttttgattggatactctcaacggaaatcagattttacagaaactatccaatca
3   ccct--tttttgattggatactctcaacggaaatcagattttagagaaactatccaatca
4   ccatagaatttgattggatgaatttcgcggaaatcagatttttgagattctacccaatca
5   atca--attttgattggatgatttgaaaggaaatcagatttttgagattttatccaatca
6   cgct--gatttgattggatgatttggaaggaaatcagattttggcgattctatccaatca
7   ccct--tttttgattggatactctcaacggaaatcagattttagagaaactatccaatca
8   ---t--t----gattggatgatttatggggaaatcagattttgagattctatccaatca
9   aaag--attttgattggatgatttgaaaggaaatcagattttgagattttatccaatca
10  cccg--catctgattggataatcgcagagaaattcagatttgtgagattttgtccaatca
11  agag--aatctgattggatgatttaagaggaaatcagatttttgggattctatccaatca
12  ccct--tttttgattggatactctcaacggaaatcagattttagagaaactatccaatca
13  agag--aatcggattggatgatttaagaggaaatcagattgtggggattctatccaatca
14  ccct--tttttgattggatactctcaatcgaaatcagattttggcgagactatccaatca
15  acct--tttttgattggatactctgaatggaaatcagattttggagaaactatccaatca
16  ---t--t----gattggatgatttgggggggaaatcaaatttttgagattctatccaatca
17  ccct--tttttgattggatactctcaacggaaatcagattttagagaaactatccaatca
18  agag--aatctgattggatgatttaagaggaaatcagatttttgggattctatccaatca
19  cgct--gatctgattggatgatttggaaggaaatcagattttggcgattctatccaatca
```

Fig. 15. (*Continued*)

```
20  ccct--tttttgattggatactctcaacggaaatcagattttagagaaactatccaatca
21  cgcg--aatgcgattggatgatttttttgaaattcggattaattaaatgctatccaatca
    ...........********...........**.**..***......*...*..*******

1   gaatgcaaatctgaattgtatttggataagatgca-a-agat-t-tt-a-a-ataattaa
2   gaatgcatatctgaattgtatttggataacatgcg-a-agat-t-tt-a-a-ataattaa
3   gaatgcatatctgaattgtatttggataacatgcg-a-agat-t-tt-a-a-ataattaa
4   gaatcaaaatcttcacaaaatttggataat-tgggga-atat-t-tt-a-a-a-ag--aa
5   gatttgagatctccgatcaatttggataat-ta-g-a-t-at-tatt-a-a-aaaa--aa
6   gaagcgagatctaaattaaatgtggatt-gaagca-atacat-t-a--a-a-agaa--aa
7   gaatgcatatctgaattgtatttggataacatgcg-a-agat-t-tt-a-a-ataattaa
8   gaatcaaaatcttaattaatttaggataat-tg-g-ata-atat-tt-a-a-aaaa--aa
9   aatttggaatctgtgagcaatttggataat-ta-g-a-t-at-tattca-a-a-aa--aa
10  gattgaatatctcaattaattgaggataag-tgtgaa-agat-t-tt-a-aga-agcgga
11  gaatcaaaatcttaattaaatttggataggagg-gta-atat-t-tt-a-a-a-ag--aa
12  gaatgcatatctgaattgtatttggataacatgcg-a-agat-t-tt-a-a-ataattaa
13  gaatcaaaatcttaattaaatttggataggagg-gta-atat-t-tt-a-a-a-ag--aa
14  gattgcatatctaaattgtatttggataagataag-a-agat-t-tt-a-a-ataattaa
15  gaatgcaaatctgaaatgtatttggataagatgcg-a-agat-t-tt-a-a-ataattaa
16  gaatcaaaatcttaattaatttaggataat-tg-g-ata-atat-tt-a-a-a-aa--aa
17  gaatgcatatctgaattgtatttggataacatgcg-a-agat-t-tt-a-a-ataattaa
18  gaatcaaaatcttaattaaatttggataggagg-gta-atat-t-tt-a-a-a-ag--aa
19  gaagcgagatctaaattaaatgtggatt-g--gcgca-atacat-ta-a-a-agaa--aa
20  gaatgcatatctgaattgtatttggataacatgcg-a-agat-t-tt-a-a-ataattaa
21  gaatcgatatctggattaattttggataatatccc-aga-at-t-ta-aga-atgat-at
    .*......****.........*..****.........*...*..*....*.*.*.......

1   ag--a-tctttgccccccagtgaatataa-tgattaaa-taaacaaata-aa-at-ta-aa
2   ag--a-tctttgccccccaatgaatataa-tgattaaa-caaacaaata-aa-aa-ta-aa
3   ag--a-tctttgccccccaatgaatataa-tgattaaa-caaacaaata-aa-aa-ta-aa
4   aggaa-tcttttcccaaaa-gactataa-tcattaaaacaaa-aaata-aaaaacta-a-
5   agaga-tcttt-ccccaaa-gactataa-tcattaaaacaaa-aa-ta-aataa-tctaa
6   agcaa-tcatt-ccccaaa-gccaataa-tcaaaaaa-caaa-aaata-aa-a--ta-ac
7   ag--a-tctttgccccccaatgaatataa-tgattaaa-canacaaata-aa-aa-ta-aa
8   aggaa-tctttaccta-aa-gcctataa-tcattaaaacaaa-aaataaaaaaacta-at
9   agaga-tcttt-ccccaaa-gactataa-tcattaaaacaaa-ata-a-aa-aa-tctaa
10  a---a-tcttcccccccatccgatataa-tcaa-aaatcaaa-a--tc-aa-aa-ta-ac
11  aggaa-tcttt-ccccaaa-ggctataa-tcattaaaacaaa-aaata-aa-aa--a-ac
```

Fig. 15. (*Continued*).

```
P41048    MQARGTVKVQGDAKVDGKMSTGQHSHHQHLNSTQANATTTALEYRAMNRPLYRGPISHNI
P41049    MQARGTVKVQGDANVDGKMSTGQHPHHQHLNSTQANATTTALEYRAMNRPLYRGPISHNI
P25027    M--KD--K---AP-V-S--S--QQDHFSR--GGAVGGKPIS-DVRGTSRPFYRKPVSHNT
P41045    M-SRS--K---E--V-S--P--NLSQQKR--G-DVRSAGIS-GF-S-S-PIYRGRLNHSA
          *......*......*.................................*.**....*..

P41048    ISEMAEGFYVLSGGYKKLFIPSKDVYALMQNVGMHLTEEEFHDALRVIGQSEPQNADELS
P41049    ISEMAEGFYVLSGGYKKLFIPSKDVYALMQNVGMHLTEEEFHDALRVIGQSEPQNADELS
P25027    IAELAEGFRVLSNGQKTISIPMKEVSALMASVGLHLSDEEFHEVMRVFGQGEQTNTEELS
P41045    SAELQEGYRILTGGQKANIISDKDLFKAIHSCGLHTSEEEVNDLLRVVHQDERT-LG-LE
          ..*..**...*..*.*...*..*.........*.*...**.....**..*.*......*.

P41048    FSDFLLLMTREVDDTMADELRSAFFHYDKYKTGYVTRKQFTELFATLGERSTPEELEELL
P41049    FSDFLLLMTREVDDTMADELRSAFFHYDKHKTGYVTRKQFTELFATLAERSTPEELEELL
P25027    FKDFLSLMMCEVDDTMLEEMRGAFLHYDKQKTGFVTKKQFTELFATGGECSTPEEVEELL
P41045    FPEFMMLMTKGIDEASIAEMRRPFSVLDKAKTGVITKKQFTELFVSSGEHSSAEELEELM
          *..*..**....*.....*.*..*...**.***..*.*******....*.*..**.***.

P41048    AVAEVDETDDKIDYNRFVNELTSRVNCM
P41049    AVAEVDETDDKIDYNRFVNELTSRVNCM
P25027    TIAEQDETDDKIDYNRFINELIHRLNFM
P41045    LLAETSEELEVVDYNKLINELAILLNKM
          ..**..*.....***...***....*.*
```

Fig. 16. Alignment result of data set PD1 by the proposed GA-ACO algorithm.

```
P08165      M----APRCWRWWPWSSWTRTR-LPP-SRSIQNFGQ-HFSTQEQTPQICVVGSGPAGFYT
Q61578      M----APRCWHWWRWSAWSGLRPSPSRSTPTPGFCQ-KFSTQEKTPQICVVGSGPAGFYT
P22570      M----ASRCWRWWGWSAWPRTR-LPP-AGSTPSFCH-HFSTQEKTPQICVVGSGPAGFYT
AAH02960    M----ASRCWRWWGWSAWPRTR-LPP-AGSTPSFCH-HFSTQEKTPQICVVGSGPAGFYT
P56522      M----APRCWRWWSWSAWPGVRPLPSRSTPTPGFCK-KFSTQETTPQICVVGSGPAGFYT
P82861      MSTHKAALC-KVQILKLFL-ISARCVRITRFYGVCGLSTSTPASSPKVCIVGGGPAGFYT
            *....*..*.............................**....*..*.**.*******

P08165      AQHLLKHHSRAHVDIYEKQLVPFGLVRFGVAPDHPEVKNVINTFTQTARSDRCAFYGNVE
Q61578      AQHLLKHHTHAHVDIYEKQLVPFGLVRFGVAPDHPEVKNVINTFTQTARSDRCAFQGNVV
P22570      AQHLLK-HPQAHVDIYEKQPVPFGLVRFGVAPDHPEVKNVINTFTQTAHSGRCAFWGNVE
AAH02960    AQHLLK-HPQAHVDIYEKQPVPFGLVRFGVAPDHPEVKNVINTFTQTAHSGRCAFWGNVE
P56522      AQHLLKHHTRAHVDIYEKQLVPFGLVRFGVAPDHPEVKNVINTFTQTARSDRCAFRGNVV
P82861      AQHLVKTRTDVQVDIYERLPVPFGLVRFGVAPDHPEVKNVINTFTQTARHARCSFHGNVR
            ****.*......*****...****************************...**.*.***.

P08165      VGRDVTVQELQDAYHAVVLSYGAEDHQALDIPGEELPGVFSARAFVGWYNGLPENRELAP
Q61578      VGRDVSVPELREAYHAVVLSYGAEDHQPLGIPGEELPGVVSARAFVGWYNGLPENQELAP
P22570      VGRDVTVPELQEAYHAVVLSYGAEDHRALEIPGEELPGVCSARAFVGWYNGLPENQELEP
AAH02960    VGRDVTVPELREAYHAVVLSYGAEDHRALEIPGEELPGVCSARAFVGWYNGLPENQELEP
P56522      VGRDVSVPELREAYHAVVLSYGAEDHQPLEIPGEELPGVVSARAFVGWYNGLPENQKLAP
P82861      VGKDVTVEELQEAYHAVVLSYGAEGNRTMGVPGEDLAGVYSAKDFVGWYNGLPRNRELRP
            **.**.*.**..***********.......***.*.**.**..*********.*..*.*

P08165      DLSCDTAVILGQGNVALDVARILLTPPDHLEKTDITEAALGALRQSRVKTVWIVGRRGPL
Q61578      DLSCDTAVILGQGNVALDVARILLTPPEHLEKTDITEAALGALRQSRVKTVWIVGRRGPL
P22570      DLSCDTAVILGQGNVALDVARILLTPPEHLERTDITKAALGVLRQSRVKTVWLVGRRGPL
AAH02960    DLSCDTAVILGQGNVALDVARILLTPPEHLERTDITKAALGVLRQSRVKTVWLVGRRGPL
P56522      DLSCDTAVILGQGNVALDVARILLTPPEHLEKTDITEVALGVLRQSRVKTVWIVGRRGPL
P82861      DLSCETAVILGQGNVALDVARMLLSPVDILKKTDITQHALDALAESSVRRVLIVGRRGPL
            ****.****************.**.*...*..****..**..*..*.*..*..*******

P08165      QVAFTIKELREMIQLPGTRPMLDPADFLGLQDRIKEAARPRKRLMELLLRTATEKPGVEE
Q61578      QVAFTIKELREMIQLPGTRPILDPSDFLGLQDRIKDVPRPRRRLTELLLRTATEKPGVEE
P22570      QVAFTIKELREMIQLPGARPILDPVDFLGLQDKIKEVPRPRKRLTELLLRTATEKPGPAE
AAH02960    QVAFTIKELREMIQLPGARPILDPVDFLGLQDKIKEVPRPRKRLTELLLRTATEKPGPAE
P56522      QVAFTIKELREMIQLPGTQPILDPSDFLGLQDRIKDVPRPRKRLTELLLRTATEKPGVEE
P82861      QVACTIKELREMVNLPGTRPEMEASEFEGIAETLKSLARPRKRLTELLLKAAIETPGEEE
            ***.********..***..*.......*.*.....*...***.**.****..*.*.**..*

P08165      AARRASASRAWGLRFFRSPQQVLPSPDGRRAAGIRLAVTRLEGIGEATRAVPTGDVEDLP
Q61578      AARQALASRAWGLRFFRSPQQVLPTPDGQRVAGIRLAVTSLEGVGESTRAVPTGDVEDLP
P22570      AARQASASRAWGLRFFRSPQQVLPSPDGRRAAGVRLAVTRLEGVDEATRAVPTGDMEDLP
AAH02960    AARQASASRAWGLRFFRSPQQVLPSPDGRRAAGVRLAVTRLEGVDEATRAVPTGDMEDLP
P56522      AARRALASRAWGLRFFRSPQQVLPTPDGRRVAGIRLAVTRLEGVGESTRAVPTGDVEDLP
P82861      LEKRNKAERAWGFRFLRRPLEVLPSTDLIRAAGIRLAVNRLEGDGEGVRAVATGEVEDVE
            ......*.****.**.*.*..***..*..*.**.****..***..*..***.**..**..
```

Fig. 17. Alignment result of data set PD2 by the proposed GA-ACO algorithm.

```
P08165      CGLVLSSIGYKSRPIDPSVPFDPKLGVVPNMEGRVVDVPGLYCSGWVKRGPTGVITTTMT
Q61578      CGLLLSSVGYKSRPIDPSVPFDPKLGVIPNTEGRVVNVPGLYCSGWVKRGPTGVITTTMT
P22570      CGLVLSSIGYKSRPVDPSVPFDSKLGVIPNVEGRVMDVPGLYCSGWVKRGPTGVIATTMT
AAH02960    CGLVLSSIGYKSRPVDPSVPFDSKLGVIPNVEGRVMDVPGLYCSGWVKRGPTGVIATTMT
P56522      CGLLLSSVGYKSRPIDPSVPFDPKLGIIPNTEGRVVNAPGLYCSGWVKRGPTGVITTTMT
P82861      CGLVISSIGYKSLPIDPAVPFDGRKAIIPNTMGRVQQTAGLYCSGWVKRGPTGVIATTMN
            ***..**.****.*.**.****.......**..***....***************.***.

P08165      DSFLTGQILLQDL-KAGHLPSGPRPGSAFIKALLDSRGVWPVSFSDWEKLDAEEVSRGQA
Q61578      DSFLTSQALLEDL-KAGLLPSGPRPGYVAIQALLSNRGVRPVSFSDWEKLDAEEVSRGQG
P22570      DSFLTGQMLLQDL-KAGLLPSGPRPGYAAIQALLSSRGVRPVSFSDWEKLDAEEVARGQG
AAH02960    DSFLTGQMLLQDL-KAGLLPSGPRPGYAAIQALLSSRGVRPVSFSDWEKLDAEEVARGQG
P56522      DSFLTSQVLLKDL-KAGLLPSGPRPGYTAIQALLSDRGVRPVSFSDWEKLDAEEVARGQG
P82861      DSFDTARTLLQDIGKGTLDVSSVKPGSQGVSALLEKRGVKPVSFSDWEKIDSEETRRGET
            ***.*...**.*..*.....*...**.....***..***.*********.*.**..**..

P08165      SGKPREKLLDPQEMLRLLGH
Q61578      TGKPREKLVDRREMLRLLGH
P22570      TGKPREKLVDPQEMLRLLGH
AAH02960    TGKPREKLVDPQEMLRLLGH
P56522      TGKPREKLVDRREMLQLLGH
P82861      RGKPREKMLDVGEMLQVARA
            .******..*..***.....
```

Fig. 17. (*Continued*).

```
P00978      MRSLSGLLLLLTACLAVNASSVPTLPDDIQVQENFDLSRIYGKWFNVAVGSTCPWLKRFK
P04366      --AVSA-SPVLT--L-----P-N---D-IQVQENFDLSRIYGKWFHVAVGSTCPWLKRFK
P02760      MRSLGALLLLLSACLAVSAGPVPTPPDNIQVQENFNISRIYGKWYNLAIGSTCPWLKKIM
O70160      MWGAEALLLLLGSCLAISVNPVLTLPDDIQVQENFDESRMYGKWYSLAIGSTCPWLKRIK
Q60559      MLGLGTLFLLLAACPASRADPVPALPD-IQVQENFNESRIYGKWFNLAVGSTCPWLSRIK
Q64240      MQGLGALFLLLTACLTLKADNVPTLPD-IQVQENFNEARIYGKWFNLAVGSTCPWLRRIK
Q62577      MKGLGTLFLLLTACLASRADDAATLPD-IQVQENFIESLIYGKWFNLAVGSTCPWLRRIK
AAD01995    MQGLRTLFLLLTACLASRADPASTLPD-IQVQENFSESRIYGKWYNLAVGSTCPWLSRIK
Q07456      MQGLRTLFLLLTACLASRADPASTLPD-IQVQENFSESRIYGKWYNLAVGSTCPWLSRIK
Q9DBJ9      MQGLRTLFLLLTACLASRADPASTLPD-IQVQENFSESRIYGKWYNLAVGSTCPWLSRIK
            ..........*..............*.*******.....****...*.*******....

P00978      EKMTMSTVVLIAGPTSKEISVTNTHRRKGVCESISGTYEKTSADGKFLYHKAKWNITMES
P04366      DKMTMGTLMLGEGATEREISVTKTHRRKGICEVISGAYEKTSTDGKFLYHKSKWNITMES
P02760      DRMTVSTLVLGEGATEAEISMTSTRWRKGVCEETSGAYEKTDTDGKFLYHKSKWNITMES
O70160      DRLSVDTMVLGKGATEAQMSTTRTHWRRGVCERTSGFYEKTDTAGKYLYSNPKWNLTMES
Q60559      NKMSMSTLVLREGATGAEISTTSTRWRRGVCEEVSGTYEKTDMDGKFLYHKSKWNVTLES
Q64240      NKMSVSTLVLQEGATEAEISVTSTQWRKGVCEEISGVYQKTDIDGKFLYHKSKWNATLES

Q62577      DKMSMSTLVLQEGATEAEISMTSTRWRRGVCEEISGAYEKTDIDGKFLYHKSKWNITLET
AAD01995    DKMSVSTLVLQEGATETEISMTSTRWRRGVCEEITGAYQKTDIDGKFLYHKSKWNITLES
Q07456      DKMSVQTLVLQEGATETEISMTSTRWRRGVCEEITGAYQKTDIDGKFLYHKSKWNITLES
Q9DBJ9      DKMSVSTLVLQEGATETEISMTSTRWRRGVCEEITGAYQKTDIDGKFLYHKSKWNITLES
            ......*..*..*.*....*.*.*..*.*.**...*.*.**...**.**...***.*.*.

P00978      YVVHTNYDEYAIFLTKKLSRRHGPTITVKLYGREPQLRESLLEEFREVALGVGIPEDAI-
P04366      YVVHTNYDEYAIFLTKKFSRRHGPTLTAKLYGREPQLRESLLEEFREVALGVGIPEDSI-
P02760      YVVHTNYDEYAIFLTKKFSRHHGPTITAKLYGRAPQLRETLLQDFRVVAQGVGIPEDSI-
O70160      YVVHTNYDEYATVLTKKFSRHHGPTITAKLYGREPQLRDSLLQGFREMALSVGIPEDSI-
Q60559      YVVHTNYDEYAIFLTKKFSRHHGPTITAKLYGREPQLRDSLLQEFREVALSVGIPENSIV
Q64240      YVVHTNYDEYAIFLTKKFSHRHGPTITAKLYGREPQLRDSLLQEFREVALSVGIPENSIV
Q62577      YVVHTNYDEYAIFLTKKFSHYHGPTITLKLYGREPKLRDSLLLEFREVALSMGIPENSIV
AAD01995    YVVHTNYDEYAIFLTKKSSHHHGLTITAKLYGREPQLRDSLLQEFKDVALNVGISENSII
Q07456      YVVHTNYDEYAIFLTKKSSHHHGLTITAKLYGREPQLRDSLLQEFKDVALNVGISENSII
Q9DBJ9      YVVHTNYDEYAIFLTKKSSHHHGLTITAKLYGREPQLRDSLLQEFKDVALNVGISENSII
            ***********..****.*..**.*.*.*****.*.**..**..*...*...**.*..*.
```

Fig. 18. Alignment result of data set PD3 by the proposed GA-ACO algorithm.

```
P00978     FTMPDRGECVPGEQDPVPTPLSRARRAVLTQEEEGSGAGQPVTNFSKKADSCQLDYSQGP
P04366     FTMPDRGECVPGEQEPEPTLLSRARRAVLPQEEEGSGAGQPVADFSKKEDSCQLGYSQGP
P02760     FTMADRGECVPGEQEPEPILIPRVRRAVLPQEEEGSGGGQLVTEVTKKEDSCQLGYSAGP
O70160     FTMANRGECIPGEQIPQPTPALRARRAVLLQEDEGSGAGQPVTDFTKKEDSCQLGHSAGP
Q60559     F-MEDRGECVPGDLEQKSTSLLRARRAVLPQENEGSGTGPLVTDVLKKEDSCQLSYSEGP
Q64240     F-MADRGECVPGDREVESTSFARARRAVLPQENEGSGSEPLITGTLKKEDSCQLNYSEGP
Q62577     F-MADKGECVPGDQEVKPSPHLRARRAVLPQENEGSGSEPLVTGILKKEDSCQLTYSEGP
AAD01995   F-MPDRGECVPGDREVEPTSIARARRAVLPQESEGSGTEPLITGTLKKEDSCQLNYSEGP
Q07456     F-MPDRGECVPGDREVEPTSIARARRAVLPQESEGSGTEPLITGTLKKEDSCQLNYSEGP
Q9DBJ9     F-MPDREECVPGDREVEPTSIARARRAVLPQESEGSGTEPLITGTLKKEDSCQLNYSEGP
           *.*....**.**..........*.*****.**.****.........**.*****..*.**

P00978     CLGLFKRYFYNGTSMACETFLYGGCMGNGNNFLSEKECLQTCRTVEACNLPIVQGPCRSY
P04366     CLGMIKRYFYNGSSMACETFHYGGCMGNGNNFVSEKECLQTCRTVEACSLPIVSGPCRGF
P02760     CMGMTSRYFYNGTSMACETFQYGGCMGNGNNFVTEKECLQTCRTVAACNLPIVRGPCRAF
O70160     CLGLFKRYFYNGSSMACEIFHYGGCLGNGNNFNSEKECLQTCRTVAACNLPIVPGPCQGS
Q60559     CLGMIEKYYYNGASMACETFHYGGCLGNGNNFNSEKECLQTCRTVAACSLPIVQGPCRAY
Q64240     CLGMQQKYYYNGASMACETFQYGGCLGNGNNFASEKECLQTCRTIAACNLPIVQGPCRAF
Q62577     CLGMMERYHYNGTSMACETFQYGGCLGNGNNFISEKECLQTCRTVAACNLPIVQGPCRAY
AAD01995   CLGMQERYYYNGASMACETFQYGGCLGNGNNFISEKDCLQTCRTIAACNLPIVQGPCRAF
Q07456     CLGMQERYYYNGASMACETFQYGGCLGNGNNFISEKDCLQTCRTIAACNLPIVQGPCRAF
Q9DBJ9     CLGMQERYYYNGASMACETFQYGGCLGNGNNFISEKDCLQTCRTIAACNLPIVQGPCRAF
           *.*....*.***.*****.*.****.******..**.*******..**.****.***...

P00978     IQLWAFDAVKGKCVRFSYGGCKGNGNKFYSEKECKEYCGIPGEADEELLRFSN
P04366     FQLWAFDAVQGKCVLFNYGGCQGNGNQFYSEKECKEYCGVPGEEDEELLRSSN
P02760     IQLWAFDAVKGKCVLFPYGGCQGNGNKFYSEKECREYCGVPGDGDEELLRFSN
O70160     AQLWAFDAIKGKCVRFTYGGCQGNGNKFYSEKECKEYCGAPGDGDEELLRFSN
Q60559     VELWAFDAAQGKCVQFSYGGCKGNGNKFYSEKECKEYCGVPGDGYEELTR-S-
Q64240     AELWAFDAAQGKCIQFIYGGCKGNGNKFYSEKECKEYCGVPGDGYEELTR-S-
Q62577     IKLWAFDAAQGKCIQFTYGGCKGNGNKFYSEKECKEYCGVPGDG-DK-T--N-
AAD01995   IKLWAFDAAQGKCIQFHYGGCKGNGNKFYSEKECKEYCGVPGDGYEELIR-S-
Q07456     IKLWAFDAAQGKCIQFHYGGCKGNGNKFYSEKECKEYCGVPGDGYEELIR-S-
Q9DBJ9     IKLWAFDAAQGKCIQFHYGGCKGNGNKFYSEKECKEYCGVPGDGYEELIR-S-
           ..******..***..*.****.****.*******.****.**..........
```

Fig. 18. (*Continued*).

```
P02212    -P---SVQ-G-AAAQ-LTADV--KKD-LRDSWKVI-GSDKKGNGVALMTTLFADNQETIG
P02213    -P---SVY-D-AAAQ-LTADV--KKD-LRDSWKVI-GSDKKGNGVALMTTLFADNQETIG
Q26505    MP---SVY-D-AAAQ-LTADV--KKD-LRDSWKVI-GSDKKGNGVALMTTLFADNQETIG
P14394    -P---SVQ-D-AAAQ-LTADV--KKD-LRDSWKVL-GSDKKGDGMALMTTLFNDHQETIA
P04251    -S---TVA-E-LANA-VVSNAD-QKDLLRLSWGVL-SVDMEGTGLMLMANLFKTSSAART
O02480    MS---KVA-E-LANA-VVSNAD-QKDLLRMSWGVL-SVDMEGTGLMLMANLFKTSPSAKG
P14822    -S---KVA-E-LANA-VVSNAD-QKDLLRMSWGVL-SVDMEGTGLMLMANLFKTSPSAKG
Q17156    -M---SFE-E-AALE-VTGSEK-IKEDLRLTWGIL-SNELEDTGVTLMLTLFKMEPGSKA
P25165    -S---TFG-E-LANE-VVNNSY-HKDLLRLSWGVL-SDDMEGTGLMLMANLFNMSPESRL
P14821    ------VA-D-AVAK-VCGSEA-IKANLRRSWGVL-SADIEATGLMLMSNLFTLRPDTKT
Q17155    MADEEQIAKAVFKEAKALCDSDAAKENLRSSFKIL-ANDIESSGLQLMKNLFTSHPDTLG
P14395    ------VA-D-AVAK-VCGSEA-IKGNLRRSWGVLMSADIEATGLTYLANLFTLRPDTKT
Q26269    -P---SVA-A-AVSA-VTNKD--VAQEIKDTWPFL-KKDMVANGISLMIKMFEKTPESLE
AAB24577  -N---K---D-VAQE-I-------KD----TWPFL-KKDMVANGISLMIKMFEKTPESLE
Q17157    -M---SVA-K-AIED-ATQPG--NRNLIRESWNMM-AADRK-NGVELMRLLFEMAPESKK
          ........................................*.......*........

P02212    YFKRLGNVSQGM-A-NDKLRGHSITLMYALQNFIDQLDNTDDLVCVVEKFAVNHITRKIS
P02213    YFKRLGDVSQGM-A-NDKLRGHSITLMYALQNFIDQLDNPDDLVCVVEKFAVNHITRKIS
Q26505    YFKRLGDVSQGM-A-NDKLRGHSITLMYALQNFIDQLDNPDDLVCVVEKFAVNHITRKIS
P14394    YFKRMGDVSQGM-A-NSKLRGHSITLMYALQNFIDQLDSTDDLICVVEKFAVNHITRKIS
P04251    KFARLGDVSAGK-D-NSKLRGHSITLMYALQNFIDALDNVDRLKCVVEKFAVNHINRQIS
O02480    KFARLGDVSAGK-D-NSKLRGHSITLMYALQNFVDALDDVERLKCVVEKFAVNHINRQIS
P14822    KFARLGDVSAGK-D-NSKLRGHSITLMYALQNFVDALDDVERLKCVVEKFAVNHINRQIS
Q17156    RFGRFGNIDSGM-G-RDKLRGHSITLMYALQNFMDSLDNTEKLRCVVDKFAVNHRIRKIS
P25165    KFGRLGHLSTGR-D-NSKLRGHSITLMYALKNFVDALDDVDRLKCVVEKFAVNHINRQIS
P14821    YFTRLGDVQKGK-A-NSKLRGHAITLTYALNNFVDSLDDPSRLKCVVEKFAVNHINRKIS
Q17155    YFARFKDVSLGK-Q-NWQLRGHAITLMYALMNFVDALDEPDRLKCVVLKFAVNHNNRSIS
P14395    YFTRLGDVQKGK-A-NSKLRGHAITLTYALDWFVDSLDDPSRLKCVVEKFAVNHINRKIS
Q26269    FFKRMGDLSNPK-A-NRKLRGHGSSLMYGIMAFVEQLDSPSDLTDVAGKFAFNHVNRNIS
AAB24577  FFKRMGDLSNPK-A-NRKLRGHGSSLMYGIMAFVEQLDSPSDLTDVAGKFAFNHVNRNIS
Q17157    EFTRLGDISPANIPYNRKLNGHGITLWYALMSFVDHLDSPNDLEDVCRKFAVNHVARNVL
          .*.*.............*.**...*.*....*...**....*..*..***.**..*...

P02212    AAEFGKINGPIKKVLASK---N-FGDKYANAWAKLVAVVQAAL-----
P02213    AAEFGKINGPIKKVLASK---N-FGDKYANAWAKLVAVVQAAL-----
Q26505    AAEFGKINGPNKKVLASK---N-FGDKYANAWAKLVAVVQAAL-----
P14394    GAEFGKINGPMKKVLASK---N-FGDKYANAWAKLVGVVQAAL-----
P04251    ADEFGEIVGPLRQTLKARMG-SYFDEDTVSAWASLVAVVQAAL-----
O02480    ADEFGEIVGPLRQTLKARMG-NYFDEDTVAAWASLVAVVQAAL-----
```

Fig. 19. Alignment result of data set PD4 by the proposed GA-ACO algorithm.

```
P14822    ADEFGEIVGPLRQTLKARMG-NYFDEDTVSAWASLVAVVQASL-----
Q17156    ASEFGWIMKPIREVLMERMG-QFYDPSFVDAWGKLIGVVQASL-AREQ
P25165    AEEFGKIVGPFRAVLRIRMG-DYFDEEIVAAWAALIAVVQAAL-----
P14821    GDAFGAIVEPMKETLKARMG-NYYSDDVAGAWAALVGVVQAAL-----
Q17155    PQVFGKINGPMDLLLKQRMG-KYYNRETANAWKQLVGVVQAAL-TTPN
P14395    GDAFGSIIEPMKETLKARMG-SYYSDDVAGAWAALIGVVQAAL-----
Q26269    SFEFQWALVPLLEVLRERLGRNRYRQETEEAWTKLVSVIQATLDDRKR
AAB24577  SFEFQWALVPLLEVLRERLGRNRYRQETEEAWTKLVSVIQATLDDRKR
Q17157    DDKFAWIKKPLEALLKNK-C-NC-KQDVVNAWCKLIDVICAVL--REG
          ...*.....*....*..............**..*..*..*.*.....
```

Fig. 19. (*Continued*).

References

[1] B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, J. Watson, Molecular Biology of the Cell, Garland Publishing, Inc., 1994.

[2] T. Jiang, L. Wang, On the complexity of multiple sequence alignment, J. Comput. Biol. 1 (1994) 337–378.

[3] J.D. Thompson, J.C. Thierry, O. Poch, RASCAL: rapid scanning and correction of multiple sequence alignments, Bioinformatics 19 (9) (2003) 1155–1162.

[4] M.A. McClure, T.K. Vasi, W.M. Fitch, Comparative analysis of multiple protein sequence alignment methods, Mol. Biol. Evol. 11 (4) (1994) 571–592.

[5] J.D. Thompson, F. Plewniak, O. Poch, A comprehensive comparison of multiple sequence alignment programs, Nucl. Acids Res. 27 (1999) 2682–2690.

[6] D.F. Feng, R.F. Doolittle, Progressive sequence alignment as a prerequisite to correct phylogenetic trees, J. Mol. Evol. 25 (1987) 351–360.

[7] J.D. Thompson, D.G. Higgins, 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 (1994) 4673–4680.

[8] Y. Wang, K.-B. Li, An adaptive and iterative algorithm for refining multiple sequence alignment, Comput. Biol. Chem. 28 (2004) 141–148.

[9] H. Carrillo, D.J. Lipman, The multiple sequence alignment problem in biology, SIAM J. Appl. Math. 48 (1998) 1073–1082.

[10] D.J. Lipman, S.F. Altschul, J.D. Kececioglu, A tool for multiple sequence alignment, Proc. Natl. Acad. Sci. USA 86 (1998) 4412–4415.

[11] J. Stoye, V. Moulton, A.W. Dress, DCA: an efficient implementation of the divide-and-conquer approach to simultaneous multiple sequence alignment, Comput. Applic. Biosci. 13 (6) (1997) 625–626.

[12] J. Stoye, Multiple sequence alignment with the divide-and-conquer method, Gene 211 (2) (1998) 45–56.

[13] E.W. Myers, W. Miller, Multiple sequence alignment using simulated annealing, Comput. Applic. Biosci. 4 (1) (1988) 11–17.

[14] M. Isokawa, M. Wayama, T. Shimizu, Multiple sequence alignment using a genetic algorithm, Genome Inform. 7 (1996) 176–177.

[15] C. Notredame, D.G. Higgins, SAGA: sequence alignment by genetic algorithm, Nucl. Acids Res. 24 (8) (1996) 1515–1524.

[16] C. Notredame, E.A. O'Brien, D.G. Higgins, Raga: RNA sequence alignment by genetic algorithm, Nucl. Acids Res. 25 (22) (1997) 4570–4580.

[17] C. Zhang, A.K.C. Wong, A genetic algorithm for multiple molecular sequence alignment, Comput. Applic. Biosci. 13 (6) (1997) 565–581.

[18] L. Jiao, L. Wong, Novel genetic algorithm based on immunity, IEEE Trans. Syst., Man Cyber.—Part A 30 (5) (2000) 552–561.

[19] L. Cai, D. Juedes, E. Liakhovitch, Evolutionary computation techniques for multiple sequence alignment, Proc. 2000 IEEE Congress Evol. Comput. 2 (2000) 829–835.

[20] K. Chellapilla, G.B. Fogel, Multiple sequence alignment using evolutionary programming, Proc. 1999 IEEE Congress Evol. Comput. 1 (1999) 445–452.

[21] R. Thomsen, G.B. Fogel, T. Krink, A Clustal alignment improver using evolutionary algorithms, Proc. 2002 IEEE Congress Evol. Comput. 1 (2002) 121–126.

[22] S.F. Altschul, Cap casts for multiple sequence alignment, J. Theor. Biol. 138 (1989) 297–309.

[23] S.F. Altschul, R.J. Carroll, D.J. Lipman, Weights for data related by a tree, J. Mol. Biol. 207 (1989) 647–653.

[24] C. Lawrence, S.F. Altschul, M. Boguski, J. Liu, A. Neuwald, J. Wooton, Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment, Science 262 (1993) 208–214.

[25] D.E. Goldberg, Genetic Algorithm in Search, Optimization, and Machine Learning, Addison-Wesley, New York, 1989.

[26] A. Kolen, E. Pesch, Genetic local search in combinatorial optimization, Discr. Appl. Math. Combin. Oper. Res. Comput. Sci. 48 (1994) 273–284.

[27] N.L.J. Ulder, E.H.L. Aarts, H.J. Bandelt, P.J.M. Van Laarhoven, E. Pesch, Genetic local search algorithms for the traveling salesman problem, Parallel Problem solving from nature, in: Schwefel, Männer (Eds.), Proceedings of the 1st Workshop, PPSN I, vol. 496, 1991, pp. 109–116.

[28] N.J. Radcliffe, P.D. Surry, Formal memetic algorithms, in: Evolutionary Computing: Selected Papers from the AISB Workshop, 1994, 1–16.

[29] P. Moscato, M.G. Norman, A memetic approach for the traveling salesman problem, in: Parallel Computing and Transporter Applications, IOS Press, Amsterdam, 1992, pp. 187–194.

[30] E.K. Burke, A.J. Smith, Hybrid evolutionary techniques for the maintenance scheduling problem, IEEE Trans. Power Syst. 15 (2000) 122–128.

[31] J. Miller, W. Potter, R. Gandham, C. Lapena, An evaluation of local improvement operators for genetic algorithms, IEEE Trans. Syst., Man Cyber. 23 (5) (1993) 1340–1341.

[32] P. Merz, B. Freisleben, Fitness landscape analysis and memetic algorithms for quadratic assignment problem, IEEE Trans. Evol. Comput. 4 (4) (2000) 337–352.

[33] Z.-J. Lee, S.-F. Su, C.-Y. Lee, A genetic algorithm with domain knowledge for weapon-target assignment problems, J. Chin. Inst. Eng. 25 (3) (2003) 287–295.

[34] Z.-J. Lee, S.-F. Su, C.-Y. Lee, Efficiently solving general weapon-target assignment problem by genetic algorithms with greedy eugenics, IEEE Trans. Syst., Man Cyber.—Part B 33 (2003) 113–121.

[35] M. Gen, R. Cheng, Genetic Algorithms and Engineering Design, John Wiley & Sons Inc., 1997.

[36] M. Dorigo, G.D. Caro, Ant colony optimization: a new meta-heuristic, Proc. 1999 Congress Evol. Comput. 2 (1999) 1470–1477.

[37] M. Dorigo, T. Stützle, Ant Colony Optimization, The MIT Press, 2004.

[38] Z.-J. Lee, C.-Y. Lee, S.-F. Su, An immunity based ant colony optimization algorithm for solving weapon-target assignment problem, Appl. Soft Comput. 2 (2002) 39–47.

[39] C. Notredame, L. Holm, D. Higgins, T-COFFEE: a novel method for fast and accurate multiple sequence alignments, J. Mol. Evol. 302 (2000) 205–217.

[40] J.-T. Horng, C.-M. Lin, B.-H. Yang, C.-Y. Kao, A genetic algorithm for multiple sequence alignment, in: Proceedings of the GCB, 2001.