Multiple Sequence Alignment using GA and NN

Shuting Wu [1], Malrey Lee[1], YongSeok Lee[1] and Thomas M Gatton[2]

[1]*The Research Center of Industrial Technology, School of Electronics & Information Engineering, ChonBuk National University, 664-14, 1Ga, DeokJin-Dong, JeonJu, ChonBuk, 561-756, South Korea, mrlee@chonbuk.ac.kr(corresponding author)*
[2]*The School of Engineering and Technology, National University, 11255 North Torrey Pines Road, La Jolla, CA 92037 USA; E-mail: tgatton@nu.edu*

Abstract. Multiple sequence alignment (MSA) is an important tool in biological analysis. However, it is difficult to solve this class of problems, due to their exponential complexity. This paper presents an algorithm combining the genetic algorithm and a self-organizing neural network for solution to MSA. This approach demonstrates improved performance in long DNA and RNA data sets exhibiting small similarity.

1 Introduction

Multiple sequence alignment (MSA) is a pre-processing tool in the subsequent analyses of protein families. .It is has been identified as one of the challenging tasks in bioinformatics.[1] In evolutionary processing, molecular sequences may have several mutations, such as insertion, deletion and substitution.. The importance of these methods continues to increase with the exponential growth of sequence databases. Multiple sequence alignment allows comparison of the structural relationships between sequences by simultaneously aligning multiple sequences and constructing connections between the elements in different sequences. The input set of query sequences, are assumed to have an evolutionary relationship. The main problem in MSA is its exponential complexity with the considered input data set. These alignments may be used to identify profiles or hidden models that may be used to acquire knowledge for distantly related members of the family sequences, newly discovered sequences, and existing sequence databases [2]. Molecular sequences of DNA, RNA or protein are composed of several kinds of elements. DNA and RNA sequences are composed of 4 kinds of basic symbols, and protein is composed of 20.

The most widely used approach to MSA uses a heuristic search that builds the result by combining pair wise alignment, such as that used in ClustalW and T-coffee. Its main advantage is that when the errors are made at any step, the error will be propagated to the final result. This is a critical problem if more sequences are added to the alignment process. One approach in addressing this problem is to use an extension of dynamic programming for simultaneously aligning multiple sequences [3, 4]. These algorithms often have higher quality solutions, as compared to a progressive approach. The disadvantage of those algorithms is the complexity

increase and additional running time and memory requirements. This allows their use in problems with a limited number of sequences

. Other iterative approaches include simulated annealing, genetic algorithms and evolutionary programming [5-9]. Simulated annealing can be slow, but woks well to improve alignment. The genetic algorithm is applied with bit matrices, but may cause degeneracy in search performance, with poorly designed operators. Evolutionary programming works well with highly similar blocks. Gibbs sampling based on the iteration idea performs well in finding local multiple alignment blocks with no gaps [10].

Recently, genetic algorithms with local searching have been considered as a good solution for this type of problem to avoid local optimization [11-15]. The local search identifies a better applicant before the next iteration. This paper presents a generic algorithm with a self-organizing neural network to classify blocks exiting from MSA. It has the advantage of genetic algorithm, as well as the capability of finding possible solutions without premature convergence [16]. The remainder of this paper is organized in three sections. The next section describes genetic algorithms and self-organizing neural network. Next, the proposed GA-SNN algorithm for application to MSA is presented. Finally, the simulation results are presented and compared to other, existing algorithm and the conclusions of the paper are made.

2. GENETIC ALGORITHMS AND SELF-ORGANIZING NEURAL NETWORKS

Genetic algorithms [14] are population-based algorithms based on the concept of biological evolution and biological genetics [17]. When applied to optimization specific problems, genetic algorithms provide the advantage of hybridization with domain-dependent heuristics. Genetic algorithms use chromosomes to represent a possible solution of the problems, and begin with a randomly selected population set, which represents the genes in a chromosome, coded as variables of the problem. These populations produce other generations following the principles of natural selection. These include crossover, mutation over many generations and survival of the fittest. Usually, fitness values can be evaluated with some utility measure or benefit that is appropriate. Individuals with the highest survival rate have a relatively large number of offspring and an increasing numbers of individuals in each survival population can be generated with the genes from highly fit individuals. Different ancestors can produce super offspring.

A. *Self-Organizing Neural Networks*

A motif is a short length string with a high similarity that forms a highly constrained sequence. When a population is introduced to the self-organizing neural network system, those MSA classify as a motif, but include a space symbol. A better population can be generated by reorganizing the original population according those motif results as masks [20,21].

The structure of the self-organizing neural network, shown in Figure 1, contains two layers, an input layer and an output layer. Specially, all of the sub-networks share the same input layer, and the output layers have connections to the neighborhood sub-network. The number of output neurons of a sub-network is equal to the number of categories classified by this sub-network. The input neurons depend on the number of patterns, and the patterns are obtained from the given sequences. When the encoding the initial network pattern, both similarity and consecutiveness are considered. Each output category is represented by the connection weights from all input neurons. In this approach, the initialization of network and the connection of output neurons is performed through a binary tree. The top level of network, which is the first sub-network, performs classifications by dividing the input space into 3 categories. The second sub-network typically divides the input space into categories $((3\text{-}1)^2 + 1)$ and places them at the next level. The last sub-network is also the judgment level, and classifies all input as either a motif or a non-motif. If the categories have been created, the output neuron at each level, except for the top level, contains the patterns of output neurons belonging to the neuron in last level. This process avoids local convergence and allows generation of super-fit offspring.

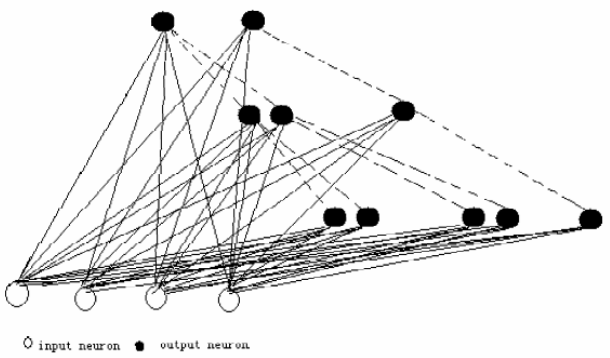

Fig. 1. The structure of self-organizing neuron network

3 THE PROPOSED GA-SNN ALGORITHM

3.1 Representation and Population Initialization

It is considered that there are M sequences to be aligned with various lengths from m_1 to m_k. DNA,RNA and protein are made of four different letters and protein are made of 20 different letters, called basic symbols. A chromosome is a matrix with fixed lengths and represented as sequences with spaces [17, 20]. A chromosome is encoded as, in Isokawa et al. (1996) and Wayama et al (1995), as a bit matrix consisting of 0's and 1's, respectively representing basic and space symbols [23,24]. The proposed representation is like the bit matrix, but only the space symbol is recorede. The chromosome X of an alignment that is composed of sequences *X* l .....Xm,can be presented in the matrix:

$$X_1.....X_m = (x_{11}, x_{12}.....x_{1m_1}), (x_{21}, x_{22}.....x_{2m_2}).....(x_{i1}, x_{i2}.....x_{im_i})$$

(1)

where *(XiI, Xi2 .....ximi)* is a number-string represented for the i - *th* sequences, and the alignment; mi, is the number of sequential symbols. Those sequences can be in alignment and, according to the definition this alignment the sequence:

*(-G - -TCGAC-), (AT - A - -*G - *C-),* (C - *CA - - - A - -*), *(A - - - TTA - G-), (GTT - TGT - TT)*

can be represented as:

(0,2,3,9),(2,4,5,7,9),(1,4,5,6,8,9),(1,2,3,7,9),(3,7)

to record the positions of space symbol.[16].

When there is m sequences to be aligned with the length (m_1, …m_i), the space ratio is denoted as *j3*. This parameter corresponds to the limit of longest length of an alignment. If the longest length among those sequences is *mmax*, the longest length of an alignment *N* should be equal to $m_{max} \times \beta$ The *N* value involves the size of search space of the alignments. If the value of *N* is too small, an optimal alignment cannot be found. If the value of *N* is too large, it will cost more time to find a high quality optimal alignment. This shows the importance of the *N* value and that *j3* is chosen according to the probability distribution.

TABLE I
THE INITIALIZATION PROCEDURE

Sequences(length)	pre-mutation→position(space symbol)→sorted	Initial alignment
GTCGAC(6)	3209785614 →3209 →(0,2,3,9)	_ G_ _ TCGAC_
ATAGC(5)	9745206183→97452 →(2,4,5,7,9)	AT_ A _ _ G_ C_
CCAA(4)	6514890237→651489 →(1,4,5,6,8,9)	C_ CA_ _ _ A_ _
ATTAG(5)	3197205486→31972 →(1,2,3,7,9)	A_ _ _ TTA_ G_
GTTTGTTT(8)	7358624910→73 →(3,7)	GTT_ TGT_ TT

The first population is generated randomly in the following steps. First, the length of alignment N is calculated according to the given sequences and space ratio. Second, a random permutation is generated from the set {0,1,2....N} and each row truncates elements to fix the length at that equal to the space symbol in the corresponding row's original alignment .Finally, the positions of all rows in the matrix are filled with the space symbols. Table 1 illustrates an example of using the random initialization procedure to generate the initial population.

3.2 Fitness Evaluation

A parameter to determine which alignment will survive in the next generation is its fitness value. The fitness of an alignment M is defined as the sum of score of all pair-wise alignments. A formalization representation is shown in formulas (2) and (3), below.

$$fitness(m_i, m_j) = \sum_{1<p<N} P_{score}(m_{ip}, m_{jp}) \quad (2)$$

$$Fitness(M) = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} fitness(m_i, m_j) \quad (3)$$

The *Fitness(M)* is represented for the sum-of-pairs score of all alignments whose length is N and is at least equal to the length of the longest sequences. The score of the *ith* column is denoted as *jitness(mi,* mj),where mi and m_j are two sequences of alignment ,*Pscore* is represented as the score of a column i consisting of the two symbols m_{ip}, m_{jp}.*Pscore* will receive score as fitness from the matrix score in PAM250. This is a biased version where all values larger than 0 are used and added to the score of the space *symbol,Pscore(x, -)=-l,Pscore(-, -)*=O. The PAM250 matrix is shown in Figure.2.

3.3. Selection and Terminations

Depending on its fitness value, selection operators determine which alignment survives in the selection pool. This means that all alignments in the pool become a parent alignment in the next generation. Either the number of generations exceeding the permitted generation denoted as *gmax* or the best fitness which does not improve over a given generations is *bmax* which

Fig. 2. The PAM250 score matrix

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	-	
C	20	8	6	5	6	5	4	3	3	3	5	4	3	3	6	2	6	4	8	0	−1	C
S		10	9	9	9	9	9	8	8	7	7	8	8	6	7	5	7	5	5	6	−1	S
T			11	8	9	8	8	8	8	7	7	7	8	7	8	6	8	5	5	3	−1	T
P				14	9	7	7	7	7	8	8	8	7	6	6	5	7	3	3	2	−1	P
A					10	9	8	8	8	8	7	6	7	7	7	6	8	4	5	2	−1	A
G						13	8	9	8	7	6	5	6	5	5	4	7	3	3	1	−1	G
N							10	10	9	9	10	8	9	6	6	5	6	4	6	4	−1	N
D								12	11	10	9	7	8	5	6	4	6	2	4	1	−1	D
E									12	10	9	7	8	6	6	5	6	3	4	1	−1	E
Q										12	11	9	9	7	6	6	6	3	4	3	−1	Q
H											14	10	8	6	6	6	6	6	8	5	−1	H
R												14	11	8	6	5	6	4	4	1(	−1	R
K													13	8	6	5	6	3	4	5	−1	K
M														14	10	12	10	8	6	4	−1	M
I															13	10	12	9	7	3	−1	I
L																14	10	10	7	6	−1	L
V																	12	7	6	2	−1	V
F																		17	15	8	−1	F
Y																			18	8	−1	Y
W																				25	−1	W
-																					0	-

Fig. 3. crossover block example

the original chromosome p_1 ,p_2 and representation
p_1=(0,2,3,9),(2,4,5,7,9), (1,4,5,6,8,9), (1,2,3,7,9), (3,7)
p_2=(1,4,5,7),(0,3,6,8,9), (1,2,3,6,7,9), (3,4,6,7,8),(2,6)
select the crossover points in parent chromsomes
point=3,5,7,8,7
ifthe element in row i of p_1
it belongs to x_1
else belongs to x_2
x_1=(0,2),(2,4), (1,4,5,6), (1,2,3,7,), (3)
x_2=(3,9),(5,7,9), (8,9), (7,9), (7)
also p_2
y_1= (1),(6,8),(3,7),(3,6),(2,4,5,6)
y_2=(4,5,7,8),(6,8),(9),(8),()
combination
$child_1$=x_1y_2=(0,2,4,5,7,8),(1,4,5,6,9), (1,2,3,7,,8), (3)
$child_2$=y_1x_2=(1,3,9),(6,8,5,7,9),(3,7,8,9),(3,6,7,9),(2,4,5,6,7)

can then be set as the termination condition. The **algorithm** pseudo code of the proposed approach is:

generate the initial population *gmax* and set *bmax* value
let *nm* be multiplied population size by mutation rate
let *nc* be multiplied population size by crosser rate
while not satisfy the termination condition do
select two chromosomes x and y from population radomly
let *x'* =x and *Y'* =y
for i=l to *nm* do
mutation(x'); mutation(y');
for i=l to *nc* do
w=crossover(x' ,*y');*
addx' ,*y'* and w into selecting pool
end for
select the best top *gmax* chromosomes to replaces the original population
endwhile

3.4. Operator Crossover and Mutation

The crossover operator produces offspring from two parent chromosomes by exchanging the information of the two parent chromosomes. The probability of crossover indicates if a selected chromosome survives in the next generation and the crossover operator can occur under this condition. However, empirical studies have shown that better results are achieved by a crossover probability of between 0.65 and 0.85. Three limiting points are then selected randomly from the parent chromosomes. For example, if two chromosomes, x and y, are selected to produce two child chromosomes, the limit points are selected at random from the parent chromosome. Here, the groups defined by the limit points are called crossover blocks, and they are the two crossovers of each chromosome namely Xl, X2, YI, and Y2. The child chromosomes can be composed of multiple crossover blocks. In each child, every crossover block is made by duplicating the corresponding parent, especially in the best child where those crossover blocks exhibit good performance. The points are denoted in the sequences to be separated as Pi, as shown in Figure.3 [15,17,18] . (FIGURE 3 IS ABOVE)

The mutation operator changes the information contained in the genome of a parent according to a given probability distribution and includes two operators, MergeSpaceBlock and baseSymbolblock. In this process, an input sequence may mutate several times until better one is produced. The frequency of selection of each operator is controlled by the probability of the mutation operator. If necessary, the individual must be repaired after mutation if the offspring does not get higher score.[16] The proportion of the longest length of chromosome strings varies from 1 to (the longest length of string)//(lmax * B), and randomly selects one operator from the mutation operators to alter the chromosome. The BaseSymbolblock operator picks a space from a randomly chosen row in a parent alignment and rearranges the space positions .Fig 4 illustrates an example of this operator.

Fig. 4. BaseSymbolblock operators example
the original chromosome and representation
(the boldfaced row is the selected row at random)
-G–TCGAC- (0,2,3,9)
AT-A–G-C-(2,4,5,7,9)
C-CA—A–(1,4,5,6,8,9)
A—TTA-G-(1,2,3,7,9)
GTT-TGT-TT (3,7)
select a number of symbols in[]
and a "-" to exchange the position
original row:GTT-[TGT-T]T
after mutition:GTT[TGT-T]-T
(0,2,3,9), (2,4,5,7,9), (1,4,5,6,8,9), (1,2,3,7,9), **(6,8)**

Fig. 5. MergeSpace operators example
the original chromosome and representation
(the boldfaced row is the selected row at random)
-G–TCGAC- (0,2,3,9)
AT-A–G-C-(2,4,5,7,9)
C-CA—A–(1,4,5,6,8,9)
A—TTA-G-(1,2,3,7,9)
GTT-TGT-TT (3,7)
select two "-" to merge the position
original row:A–[-]TTA[-]G-
after mutition:A–[-][-]TTAG-
(0,2,3,9), (2,4,5,7,9), (1,4,5,6,8,9), **(1,2,3,4,9)**, (6,8)

3.5 Self-organizing NN as a local search

3.5.1 Pair-wise distance calculation

In the proposed approach, the self-organizing neural network performs as a classifier for the local search to allow the genetic algorithm to exit from any local optimum. The local search performs as a pheromone in the Ant colony algorithm, with which the new alignment is initiated. The self-organizing NN does not obtain the output directly, but serves as a reference book to record the special motifs with space symbol.

Before each sequence is introduced, it is encoded by a process wherein every sequences in this alignment is formed into a group of 3 letters. These three sub-groups are the parameters used to calculate the distance between any given vectors .Assuming the sequences is introduced with length li, this produces (li - 2) sub-groups. For example, the sequences VGWYNGLPE generates the groups VGW,GWY, WYN, YNG,NGL,GLP, and LPE. The distance of two 3-letters groups is defined as the similarity of the 2-letter subgroups. And each sequences of length m is converted into m 3-letter group, and each 3-letter group is labeled as P = (Pl, P2, P3). For example, the 2-letter subgroups VG, GW and VW are generated from VGW, and denoted as shown in (4):

$$p = \{p_1, p_2, p_3\} = \{\dot{p}_1\dot{p}_2, \dot{p}_1\dot{p}_3, \dot{p}_2\dot{p}_3\} \tag{4}$$

The distance formula between two given 3-letter is denoted as shown in (5):

$$d = \min_{j,k,l\in 1,2,3, j\neq k\neq j} \{(\mid p_1 - q_j \mid + \mid p_2 - q_k \mid + \mid p_3 - q_l \mid) + \mid (p_1 + p_2 + p_3) - (q_1 + q_2 + q_3) \mid\} \tag{5}$$

also, the distance formula of any two input P and Q is denoted as shown in (6):

$$D(P, Q) = \sum_{i=1}^{m} d_i \tag{6}$$

3.5.2. Classification step

This process is based on the winner-take-all algorithm [22]. Essentially, when a new sequences is introduced to a sub-network, the classification process involves three processing steps. First, a similarity calculation is applied to compare the new input and the output of every sub-network at each level. Then, the all top level output categories are tested. At the next level, only the best output is selected for testing and a determination if it should be kept. In lth sub-networks ,the pair-wise distance is described as in (7):

$$d_j^l = D(x^i, w_j^{l+1}), j = 1, 2, ...p_q \tag{7}$$

where *Pq* is the number of categories in *lth* sub-networks and w_j^{j+1} belongs the category q of the *lth* sub-networks as in (8), where if I is the last sub-network, and xj belongs to the category q of the *lth* sub-network:

$$w_j^{l+1} = x_j \tag{8}$$

The cutoffs, or thresholds, of the *lth* sub-network are preset and denoted as *PI*, as in (9):

$$\max_{1\leq j\leq p_q} \{d_j^l\} < p_l \tag{9}$$

where the category q defines winner in the *lth* sub-network for the new input. Next, winner authentication of the largest pair-wise distance is determined with a similarity calculation within each category that compares the threshold value. If the threshold value is greater than or equal to the largest pair-wise distance of a category , it is labeled as a winner. Finally, there is a weighting update, when the test succeeds, for *l,if l.* If it isn't the last sub-network, an update, as shown in (10), is performed for

the *lth* subnetwork, where k is the *kth* input neural,k = *1,2.....M,M* is the length of alignment.

$$w_{kq}^{l} \models \Sigma_{i=1}^{p_q^l} w_{kj}^{l+1} \tag{10}$$

TABLE II
DATE SETS

ID	Number of sequences	AvgLen(min,max)	similarity
D1	8	457(456,457)	0.98
D2	5	1780(1775,1782)	0.90
D3	7	3330(3311,3370)	0.69
D4	9	8810(8834,8785)	0.70

TABLE III
COMPARISON OF THE PROPOSED ALGORITHM ,CLUSTAL W ,HORNG'S GENETIC ALGORITHM,GENETIC ALGORITHM WITHOUT SNN

ID	the proposed algorithm	clustal W	horng's genetic algorithm	genetic algorithm without snn
D1	111194	25343	25343	25343
D2	79845	18627	18627	18627
D3	194745	151308	150580	150580
D4	572466	534044	529892	529876

If 1 IS the last subnetwork,the update shown in (11) is performed.

$$w_{kq}^{l} \models \frac{1}{p_q^l + 1} \Sigma_{i=1}^{p_q^l + 1} x_k^j \tag{11}$$

When the test is not successful for I, all sub-networks are renewed from l-th to the last subnetwork by adding categories which contain only one pattern. The weight will update as shown in (12):

$$w_{kn}^{l} = x_k, k = 1, 2....M, n = N_l + 1, \tag{12}$$

If there is more than one winner, only one is the final winner at each level. In this step , a distance test between input sequence and the random selected part of each winning categories in step l is performed. The minimum of these distances is determined and final winner is generated. This step works similarly to the WTA algorithm.

At first, the output neurons could be set with random weight. is selected based on the average pair-wise distance. The following equation is used to calculate this distance between and *kth* and *13th* current output neurons of the *i th* subnetwork for *kth* element of the input, as in (13).

$$d_q^{avg}(x_k) = \frac{1}{p_q - 1} \sum_{j=1, j \neq k}^{P_q} D(x_k, x_j) \tag{13}$$

If $d_q^{avg}(x_k) < d_q^{avg}(x_\beta)$, a new input is chosen as a replacement. If not, this step is skipped and the test performed on the remaining neurons. Without this step, the local searching parameters of the system may not be satisfiable. The target neuron is defined to one whose categories have the most patterns. After finding an acceptable pattern, they are formatted need to and recorded for easy to alignment.

4. SIMULATION AND RESULTS

A set of parameters is determined before performing the simulation are: 1) the number of chromosomes in the selection pool is 20, 2) the sequence representation rate is 0.2, 3) the crossover and mutation rate are 0.2, and 4) the probabilities of crossover ,spaces block based ,base block base operators are 0.4,0.08 and 0.08 The scoring matrix for matching of basic symbols is PAM250 and the score between the space symbol and the base ,or basic symbol, are -1,0 respectively. In the self-organizing neural network, the number of sub-network is 5. Initially, neurons of the output layer are set randomly and the input neurons depend on the input sequence. In the simulation, the data sets used in the experiment are represented as an NCBI alignment database. The data sets D1,D16 and D17 are DNA or RNA types, with a longer length of sequences. Those data sets are shown in Table.2. When the number of sequences is high, the proposed algorithm performs better than other algorithm, especially for DNA or RNA types. Table.3 shows the results of the simulation.

5. CONCLUSION

Traditional algorithms such as Clustal W are known to be very succesful when the number of average length is low and similarity is high.Zhang et.at and Jorong Tzong horng et.at developed a genetic algorithm for MSA.Their method are directly compare to Clustal W and the scoring function of both was based on the probability.Also Gonnzalaz et.al running a simulation of sequences using genetic algorithm. No single alignment procedure can be expected to construct biologically reasonable alignment in all possible situations.if we find a scoring function under all conditions could be in accordance with biology giving mathematically score to the biologically correct alignment.then it is easy to give a comparison with all alignment.It can not be expected that any giver scoring function will yet.And now the date in public databases like NCBI which conducts research in computational biology undate very soon,for the propose of comparison,we keep the data set under the same length and similarity. in this paper,we present a novel algorithm of genetic algorithm with self-organizing neural network for MSA. Self-organizing nn as local optimization like classification is embedded into genetic algorithm to keep away from local optima.

Acknowledgement

This work was supported by grant R01-2006-000-10147-0 from the Basic Research Program of the Korea Science & Engineering Foundation.

References

1. J.D. Thompson, J.e. Thierry, O. Poch, RASCAL: rapid scanning and correction of multiple sequence alignments, Bioinformatics 19 (9) (2003)
2. T. Jiang, L. Wang, On the complexity of multiple sequence alignment, J. Comput. BioI. 1 (1994) 337-378.
3. 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.

4. J. Stoye, Multiple sequence alignment with the divide-and-conquer method, Gene 211 (2) (1998) 45-56.
5. C. Notredame, E.A. O'Brien, D.G. Higgins, Raga: RNA sequence alignment by genetic algorithm, Nucl. Acids Res. 25 (22) (1997) 4570-4580.
6. C. Zhang, A.K.C. Wong, A genetic algorithm for multiple molecular sequence alignment, Comput. Applic. Biosci . 13 (6) (1997) 565-581.
7. L. Jiao, L. Wong, Novel genetic algorithm based on immunity, IEEE Trans. Syst., Man Cyber.-Part A 30 (5) (2000) 552-561.
8. L. Cai, D. Juedes, E. Liakhovitch, Evolutionary computation techniques for multiple sequence alignment, Proc. 2000 IEEE Congress Evol. Comput. 2(2000) 829-835
9. R. Thomsen, G.B. Fogel, T. Krink, A Clustal alignment improver using evolutionary algorithms, Proc. 2002 IEEE Congress Evol. Comput. 1 (2002)121-126..
10. e. Lawrence, S.P. 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
11. e. Zhang, A.K.C. Wong, A genetic algorithm for multiple molecular sequence alignment, Comput. Applic. Biosci . 13 (6) (1997) 565-581.
12. L. Jiao, L. Wong, Novel genetic algorithm based on immunity, IEEE Trans. Syst., Man Cyber.-Part A 30 (5) (2000) 552-561..
13. D.E. Goldberg, Genetic Algorithm in Search, Optimization, and Machine Learning, Addison-Wesley, New York, 1989.
14. A. Kolen, E. Pesch, Genetic local search in combinatorial optimization, Discr. Appl. Math. Combin. Oper. Res. Comput. Sci . 48 (1994) 273-284.
15. N.LJ. Ulder, E.H.L. Aarts, HJ. Bandelt, PJ.M. Van Laarhoven, E. Pesch, Genetic local search algorithms for the traveling salesman problem, Parallel Problem solving from nature, in: Schwefel, Manner (Eds.), Proceedings of the 1st Workshop, PPSN I, vol. 496, 1991, pp. 109-116.
16. Zne-Jung Lee, Shun-Feng Su, Chen-Chia Chuang, Kuan-Hung Liu.Genetic algorithm with ant colony optimization (GA-ACO) for multiple sequence alignment. Applied Soft Computing, Volume 8, Issue 1, January 2008, Pages 55-78
17. J.-T. Horng, e.-M. Lin, B.-H. Yang, e.-Y. Kao, A genetic algorithm for multiple sequence alignment, in: Proceedings of the GCB, 2001
18. M. Gen, R. Cheng, Genetic Algorithms and Engineering Design, John Wiley & Sons Inc., 1997.
19. e. Notredame, D.G. Higgins, SAGA: sequence alignment by genetic algorithm, Nucl. Acids Res. 24 (8) (1996) 1515-1524.
20. Derong Liu, Xiaoxu Xiong, Zeng-Guang Hou, Bhaskar DasGupta. Identification of motifs with insertions and deletions in protein sequences using self-organizing neural networks. Neural Networks, Volume 18, Issues 5-6, July-August 2005, Pages 835-842
21. Bailey, T. L., & Elkan, e. (1995). Unsupervised learning of multiple motifs in biopolymers using expectation maximization. Machine Learning, 21(1-2),51-80.
22. Haykin, S. (1999). Neural networks: A comprehensive foundation (2nd ed.). Upper Saddle River, NJ: Prentice Hall (pp. 443-483).
23. Isokawa M, Wayama M, Shimizu T (1996) Multiple sequence alignment using a genetic algorithm. Genome Informatics, 7 176CI77
24. Murata M, Richardson JS, Sussman JL (1985) Simultaneous comparison of three protein sequences. Proc Natl Acad Sci USA, 82: 3073C3077