International Journal of Computational Intelligence Research.
ISSN 0973-1873 Vol.3, No.2 (2007), pp. 123–130

Multiple sequence alignment using swarm intelligence

Pedro F. Rodriguez, Luis F. Niño and Oscar M. Alonso

National University of Colombia, Computer Sciences Department,
Bogotá, Colombia
pfrodriguezj@unal.edu.co, lfninov@unal.edu.co, omalonsom@unal.edu.co

Abstract: **In this work, a novel approach to multiple sequence alignment based on Particle Swarm Optimization (PSO) is introduced. The proposed approach tries to improve a sequence alignment previously obtained using Clustal X. Special representation and operators are designed to deal with the sequence alignment problem. Some experiments over families of proteins were carried out in order to test the proposed approach. In most cases, the PSO algorithm was able to improve the starting alignment obtained using Clustal X.**
Keywords: swarm intelligence, sequence alignment, bioinformatics

I. Introduction

One of the main problems in Bioinformatics is the alignment of multiple biological sequences. The resulting aligned sequences are used to construct phylogenetic trees, to find protein families, to predict secondary and tertiary structure of new sequences, and to demonstrate the homology between new sequences and existing families [1, 2].
The work in this problem has relied on computational techniques, which include dynamic programming [3], neighbor-joining algorithms [4] and genetic algorithms [5].
In this work, an algorithm to address the multiple sequences alignment problem is proposed. The algorithm is based in swarm intelligence [7]. Particularly, it uses the well-known algorithm Particle Swarm Optimization (PSO).
Although most implementations solve the problem from scratch, starting with a set of unaligned sequences, the algorithm proposed in this work begins with an alignment obtained with another algorithm, Clustal X [6], and it gradually improves the quality of such initial alignment through a process based on swarm intelligence.
The rest of this paper is organized as follows. First, the multiple sequence alignment problem is introduced in section II. Then, the PSO algorithm is summarized in section III. Subsequently, an adaptation of the PSO algorithm to the sequence alignment problem is presented in section IV, including the representation details and operators. Section V presents the experimental results of the proposed algorithm. Finally, some conclusions are devised in section VI.

II. The sequence alignment problem

Among the main data sets in Bioinformatics are biological sequences, such as DNA molecules and proteins. A DNA molecule can be viewed as a sequence of nucleotides, represented by the letters A, G, C and T. Similarly, a protein can be seen as a sequence of aminoacids.
It is frequently needed to establish the similarity of two or more biological sequences. The similarity is thus used to establish evolutionary relations among species (one sequence is transformed in the other through an evolutionary process or they have a common ancestor), or to establish functional similarities in proteins.
However, this similarity measure must take into account that an evolutionary process can insert, delete or mutate elements of the sequences. Thus, in order to highlight the similarities of the sequences it is often convenient to insert gaps in them, leading to a higher number of symbol matches.
The similarity of two aligned sequences is measured using a scoring function, which is based on a matrix that assigns a score to every pair of symbols (based on a mutation probability). Additionally, a penalization to the insertion of gaps is required in order to avoid the insertion of an excessive number of them.
The process of finding an optimum (or at least good) match between the sequences is called sequence alignment.

A. Substitution matrices

It is important to notice that the mutation of the elements in a sequence is not uniform, in other words, a symbol has a higher probability to mutate to certain other symbols. Thus, the scoring functions are usually based on matrices which assign higher values to combinations of symbols with higher probability.

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	B	Z
C	12																					
S	0	2																				
T	-2	1	3																			
P	-3	1	0	6																		
A	-2	1	1	1	2																	
G	-3	1	0	-1	1	5																
N	-4	1	0	-1	0	0	2															
D	-5	0	0	-1	1	2	2	4														
E	-5	0	0	-1	0	0	1	3	4													
Q	-5	-1	-1	0	0	-1	1	2	2	4												
H	-3	-1	-1	0	-1	-2	2	1	1	3	6											
R	-4	0	-1	0	-2	-3	0	-1	-1	1	2	6										
K	-5	0	0	-1	-1	-2	1	0	0	1	0	3	5									
M	-5	-2	-1	-2	-1	-3	-2	-3	-2	-1	-2	0	0	6								
I	-2	-1	0	-2	-1	-3	-2	-2	-2	-2	-2	-2	-2	2	5							
L	-6	-3	-2	-3	-2	-4	-3	-4	-3	-2	-2	-3	-3	4	2	6						
V	-2	-1	0	-1	0	-1	-2	-2	-2	-2	-2	-2	-2	2	4	2	4					
F	-4	-3	-3	-5	-4	-5	-4	-6	-5	-5	-2	-4	-5	0	1	2	-1	9				
Y	0	-3	-3	-5	-3	-5	-2	-4	-4	-4	0	-4	-4	-2	-1	-1	-2	7	10			
W	-8	-2	-5	-6	-6	-7	-4	-7	-7	-5	-3	2	-3	-4	-5	-2	-6	0	0	17		
B	-4	0	0	-1	0	0	2	3	2	1	1	-1	1	-2	-2	-3	-2	-5	-3	-5	2	
Z	-5	0	-1	0	0	-1	1	3	3	3	2	0	0	-2	-2	-3	-2	-5	-4	-6	2	3

Figure. 1: The PAM250 matrix. It is the most used PAM matrix and represents the mutation probabilities of sequences with 20% of equivalence.

For proteins, the most commonly used matrices are PAM (Percent Accepted Mutation) and BLOSUM (Blocks Substitution Matrix) [9].

PAM matrices, formulated by Dayhoff in 1978, are based on the concept of punctual mutation. The PAM1 matrix indicates the probability that a symbol mutates into another in one time step, and is constructed based on sequences that differ only in 1% or less. Matrices corresponding to more time steps are obtained by multiplying the PAM1 matrix by itself. In this matrix negative values means unfeasible mutations, positive values indicates probable mutations and a value of zero indicates a mutation which is perfectly random. Figure 1 shows the PAM250 matrix.

The BLOSUM matrices, proposed by Steven Henikoff y Jorja G. Henikoff, are obtained from aligned sequences from the BLOCKS database [6]. Sequences with similar identity percentages are grouped and used to calculate the mutation probability, and the mutation probability is contrasted with the probability expected from random mutation. Thus, the BLOSUM80 is obtained from a group of sequences with identity percentage of 80%. Figure 2 shows the BLOSUM62 matrix.

B. Sequence alignment algorithms

Given a set of biological sequences, informally, the sequence alignment problem consists of finding a set of suitable shifts of the sequences and locations to insert gaps in each of the sequences in order to maximize the number of symbol matches among all the sequences. The problem of aligning two sequences has been addressed using dynamic programming. Thus, there are efficient algorithms to this problem. Some examples are Needleman & Wunsch and Smith & Waterman [3].

The multiple sequence alignment is a highly computationally demanding task due to the huge size of the search space. Moreover, algorithms which work well for two sequence alignment become too computationally expensive when they are extended to multiple sequences.

Most of the approaches to the multiple sequence alignment problem are based on the progressive approach proposed by Feng and Doolittle, or modifications of it [4]. This heuristic algorithm use pairwise alignments to constructs a global alignment by aligning the two more similar sequences, and then adding the other sequences one by one. This algorithm, though fast and simple, tends to fall into local optima, and does not provide a scoring function, which is needed to rank alignments.

Other approaches use a scoring function in order to be able to rank the alignments and to find the one with the highest score. However, these algorithms are computationally expensive even for a small number of sequences [5].

There are also some non-deterministic approaches using genetic algorithms, such as SAGA [5], which was reported to find optimal alignments even in search spaces of a considerable size (more that 10 sequences).

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	B	Z	X
C	9																						
S	-1	4																					
T	-1	1	5																				
P	-3	-1	-1	7																			
A	0	1	0	-1	4																		
G	-3	0	-2	-2	0	6																	
N	-3	1	0	-2	-2	0	6																
D	-3	0	-1	-1	-2	-1	1	6															
E	-4	0	-1	-1	-1	-2	0	2	5														
Q	-3	0	-1	-1	-1	-2	0	0	2	5													
H	-3	-1	-2	-2	-2	-2	1	-1	0	0	8												
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	5											
K	-3	0	-1	-1	-1	-2	0	-1	1	1	-1	2	5										
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	5									
I	-1	-2	-1	-3	-1	-4	-3	-3	-3	-3	-3	-3	-3	1	4								
L	-1	-2	-1	-3	-1	-4	-3	-4	-3	-2	-3	-2	-2	2	2	4							
V	-1	-2	0	-2	0	-3	-3	-3	-2	-2	-3	-3	-2	1	3	1	4						
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	6					
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	2	-2	-2	-1	-1	-1	-1	3	7				
W	-2	-3	-2	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	11			
B	-3	0	-1	-2	-2	-1	2	4	1	0	0	-1	0	-3	-3	-4	-3	-3	-3	-4	4		
Z	-3	0	-1	-1	-1	-2	0	1	4	3	0	0	1	-1	-3	-3	-2	-3	-2	-3	1	4	
X	-2	0	0	-2	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-2	-1	-1	-1

Figure. 2: The BLOSUM62 matrix.

In this work, a novel approach to address the multiple sequence alignment problem based in swarm intelligence is proposed. More specifically, an algorithm based on the Particle Swarm Optimization is used. Thus, next section will be devoted to describe the PSO algorithm.

III. Particle swarm optimization

In computational intelligence, some algorithms based on the social behavior of some animals, such as ants, bees, some birds and human beings, have been developed. One of these algorithms is called Particle Swarm Optimization (PSO) [7], an optimization algorithm which is briefly described next.

The PSO algorithm is inspired in the behavior of flocks of birds looking for food or water. The birds fly keeping some distance between them, meanwhile the swarm moves together as a whole.

In bird flocks, all the birds search food sources while they follow the leader's path. When a bird observes a food source, it becomes the leader and the rest of the swarm will follow it. However, it is possible that on the way to reach a food source, one of the birds in the swarm sees a better food source, then it will become the new leader, and the rest of the swarm will follow it. This process continues till the whole flock reaches a food source.

James Kennedy and Russel Eberhart [8] proposed PSO as a model of the behavior of bird flocks in the search for food. PSO is a population based search and optimization algorithm, similar to a genetic algorithm: particles (individuals) are elements in the search space; then each particle moves towards the leader, by applying some genetic operators. This search process continues until a termination criterion is satisfied. In PSO an individual corresponds to a phenotype in a genetic algorithm, which represents a solution to a specific problem. The main idea behind PSO is the movement of particles in the direction of the leader, denoted gbest, and also keeping track of the best location where each particle has been in the past, denoted pbest. The leader is the particle, in the swarm, for which the value of an objective function is maximum (or minimum).

In general, a particle is represented as an n-dimensional vector, and the coordinates of each particle are moved towards the corresponding coordinates of the leader particle, in an amount proportional to the distance that separates them. The movement produced is then determined by a constant k and a random number in the interval [0,1].

A general outline of the PSO algorithm is shown in figure 3.

PSO ( )

1. Generate a set of initial particles
2. Determine the leader particle gbest
3. Repeat until the termination criterion is met
   a. Update every particle (p) coordinate (x) according to
   $$p.x = p.x + k * rand() * (gbest.x - p.x) + k * rand() * (p.pbest.x - p.x)$$
   b. Update pbest for every particle
   c. Determine the leader particle gbest

Figure. 3: The PSO algorithm.

When the termination criterion is met, for instance, that the swarm does not improve after several iterations, then the

leader particle is output as a solution to the problem.
Next section will present the PSO algorithm applied to the problem of multiple sequence alignment.

IV. Proposed approach

The sequence alignment problem can be considered as an optimization problem in which the objective is to maximize a scoring function. Thus, the PSO algorithm was adapted to be used with biological sequences.
In the adapted PSO algorithm, a particle represents a sequence alignment. Thereby, as the main mechanism of the PSO algorithm is the movement of the particles towards the leader, suitable operators to implement this mechanism are proposed. The general algorithms is shown in figure 4.

PSOAlign ( )
1. Generate a set of initial particles
2. Determine the leader particle gbest
3. Repeat until the termination criterion is met
a. Measure distance between gbest and every particle
b. Move every particle towards gbest
c. Determine the leader particle gbest

Figure. 4: PSO algorithm adapted to sequence alignment.

The termination criteria can be a maximum number of iterations, or a number of iterations after which the best score does not improve.
The implicit idea of the PSO algorithm is that a set of particles that are randomly sparse over a search space will progressively move to regions which will provide better solutions to the problem, until the swarm finds a solution that it can not improve anymore. In the sequence alignment problem the search space is huge and exploring the whole search space is very demanding, computationally speaking.
Therefore, in order to reduce time complexity of the algorithm, the initial set of particles is located in a region of the search space that provides good solutions to the problem. To achieve this, the initial swarm of particles is generated by applying mutation (a genetic algorithm operator) to the solution obtained by the Clustal X algorithm. The solution obtained by clustal X is also added to the swarm. Thus, he algorithm will progressively improve the quality of such solution through the PSO algorithm dynamics.
Next, some implementation details will be discussed, such as the particle representation, scoring function and the implementation of the particle movement mechanism.

A. Data representation

In general, a swarm is made up by a set of particles, and a particle of the swarm is designated as the leader (gbest). Additionally, each particle preserves a memory with its best historical location (pbest).

As mentioned above, in the adapted PSO algorithm, a particle will correspond to a sequence alignment. An alignment is then represented as a set of vectors, where each vector specifies the positions of the gaps for one of the sequences to be aligned. Thus, a coordinate of the particle corresponds to a sequence to be aligned, and is represented with a vector of size s, where s is the maximum allowed number of gaps, which may be different for each sequence. Therefore, a set of n sequences to be aligned correspond to an n-dimensional search space.
As each particle is represented by a set of gaps, it is not necessary to store the whole sequence, but only the positions where the gaps are inserted. Therefore, the sequences are stored in a different structure, and the particles only include the information about the positions in which the gaps are added.
An example of the particle representation is shown in figure 5.

Set of three sequences to be aligned:

W	G	K	V	N	V	D
W	D	K	V	N		
S	K	V	G	G	N	

Particles:

	5	6		
Leader:	5	6	8	9
	2	8	9	

	4	5		
Particle:	1	5	6	7
	2	7	9	

Alignments represented by the particles:

	W	G	K	V	-	-	N	V	D
Leader:	W	D	K	V	-	-	N	-	-
	S	-	K	V	G	G	N	-	-

	W	G	K	-	-	V	N	V	D
Particle:	-	W	D	K	-	-	-	V	N
	S	-	K	V	G	G	-	N	-

Figure. 5: Example of particle representation.

1) Particle initialization

The size of the swarm (i.e., the number of particles) is determined by the user. Additionally, the length of the alignment has a minimum value given by the length of the largest sequence, and a maximum length given, for instance, as twice the length of the largest sequence.

The initial set of particles is generated using the solution obtained from the Clustal X algorithm. That solution is mutated several times in order to generate new particles in the same region of the search space. Finally, that solution is also added to the swarm.

2) Scoring function

The global alignment score is based on the score of the alignment of each pair of sequences. Thus each sequence should be aligned with the every other sequence. In general, the score assigned to each particle (alignment) is the sum of the scores of the alignment of each pair of sequences. The score of each pair of sequences is the sum of the score assigned to the match of each pair of symbols, which is given by the substitution matrix. This matrix includes all the possible symbols, including the gap and the related penalization.

3) Distance measure

The PSO algorithm specifies that all the particles in the swarm will move towards the leader at a speed proportional to the distance between each particle and the leader. Thus, the algorithm requires that a measure of the distance between particles be specified.
The distance will be measured as the proportion of gaps that do not match in the sequences, according to the formula

$$Distance = \frac{matching\ gaps}{total\ gaps}$$

Accordingly, two gaps match if they are in the same position in the particle and represent the same position in the alignment. An example of this is given in figure 6.
A distance measure can be defined in several ways, but clearly it should reflect the similarity of the alignments.

Particles:

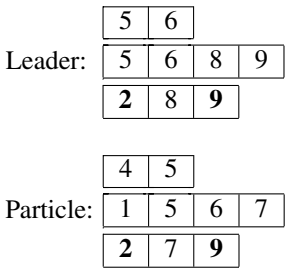
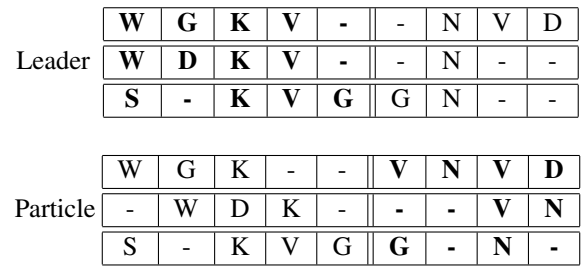


Figure. 6: Example of distance measure between particles. Matching gaps are highlighted. There are 18 gaps (9 from each particle), and 4 of them match. Thus the distance is $\frac{18-4}{18} = 0.77$. Notice that although the gap 5 in the first sequence is in the leader and in the particle, it is not in the same position and then it is not considered a match. Same happens for 5 and 6 on the second sequence.

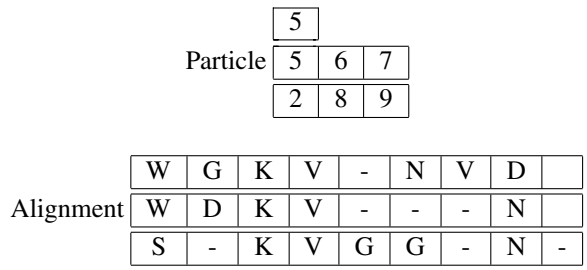
B. Operators

In the PSO algorithm each particle moves towards the leader at a speed proportional to the distance between the particle and the leader. Consequently, an operator similar to the crossover operator from genetic algorithms is proposed.
The proposed algorithm requires the selection of a crossover point which divides the alignment into two segments, and then a segment of the particle is replaced with a segment from the leader. This replacement is achieved removing from the particle the gaps that are in the segment, and then adding the gaps from the leader's segment. This can be seen in figure 7.

Before movement:



After movement:



After removing third sequence last gap:

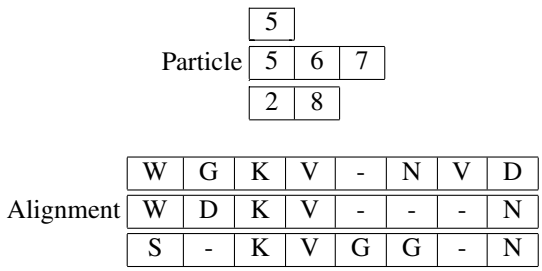


Figure. 7: Example of particle movement. The distance between particles is 0.77 and the length of the particle is 9. Thus, the crossover point must be between 1 and $0.77 * 9 = 7$. Let's suppose it is 5. As the distance is greater that 0.5, the first segment of the particle is replaced with the one of the leader. The particle is now shorter than before, because the first and second sequences became shorter and the third one ends in a gap.

The crossover point is selected at random, but it depends on the distance between the particle and the leader. The selection of the particle's segment which is replaced with the leader's segment will also depend on such distance. The distance between a particle and the leader will determine a range where the crossover point will be located, and then the exact position of the crossover point is determined by generating a random number in that range.
As the PSO algorithm specifies that the movement of the particle is proportional to the distance. Then, the distance will determine which of the two segments generated by the crossover point will be replaced with the leader's one. Thus, if the distance is greater than 0.5, the leader will provide the largest segment, otherwise the leader will provide the shorter one.
It is important to notice that after copying the segment from the leader, the aligned sequences could have different lengths, and that they are required to be of the same length. Thus, some operations have to be done to determine the exact length of the particle and the positions of the gaps. Therefore, it may be necessary to add or eliminate some gaps.
When a sequence becomes larger than it was before, it is necessary to add some gaps at the end of the other sequences. Notice that, in some cases, the final symbol of all the sequences in an alignment become gaps, therefore, they can be removed and consequently the alignment becomes shorter. This process is shown in figure 7.

Ref.	Name	Length	Identity %
1ubi	Ubiquitin	Short	< 25
1aab	High Mobility Group Protein	Short	$20 - 40$
1csp	Cold Shock Protein	Short	> 35
1havA	Hepatitis Proteinase	Medium	< 25
1ton	Tonin	Medium	$20 - 40$
1ppn	Papain	Medium	> 35
1pamA	Ciclodextrin	Large	< 25
1ac5	Carboxipeptidase	Large	$20 - 40$

Figure. 8: Protein families used in the experiments.

V. Experimental results

The proposed algorithm, called PSOAlign, was implemented in order to test its performance. The amount of memory used by the algorithm was relatively low, because most memory is

Protein family	Number of sequences	Clustal X length	Clustal X Score	PSOAlign length	PSOAlign Score	Iterations	Particles	Score Improve
1ubi	4	94	-705	94	-705	11	500	0,00%
1ubi	4	94	-705	94	-705	11	1000	0,00%
1ubi	4	94	-705	94	-705	11	2000	0,00%
1aab	4	81	57	81	62	12	500	7,00%
1aab	4	81	57	81	62	12	2000	7,00%
1csp	5	75	1393	75	1442	19	500	3,51%
1csp	5	75	1393	75	1435	16	1000	3,01%
1csp	5	75	1393	75	1450	19	2000	4,09%
1havA	5	207	-4107	207	-4100	14	500	0,17%
1havA	5	207	-4107	207	-4084	18	1000	0,53%
1havA	5	207	-4107	207	-4002	20	2000	2,55%
1havA	5	207	-4107	207	-4107	15	10000	1,36%
1ppn	5	227	4302	227	4302	11	500	0,00%
1ppn	5	227	4302	227	4306	13	5000	0,09%
1ppn	5	227	4302	227	4354	16	10000	1,20%
1ton	5	250	338	250	362	13	500	7,10%
1ton	5	250	338	250	442	17	2000	30,76%
1ton	5	250	338	250	391	17	10000	15,70%
1ac5	4	513	-737	513	-714	19	500	3,12%
1ac5	4	513	-737	513	-676	21	5000	8,22%
1ac5	4	513	-737	513	-683	19	10000	7,32%
1pamA	5	575	-8825	575	-8796	13	500	0,33%
1pamA	5	575	-8825	575	-8688	16	1000	1,56%
1pamA	5	575	-8825	575	-8718	19	5000	1,21%
1pamA	5	575	-8825	575	-8650	19	10000	1,98%

Figure. 9: Experimental results. The results obtained with PSOAlign are compared to the ones obtained with Clustal X. Only in few cases PSOAlign was not able to improve the alignment obtained with Clustal X.

used to keep track of the particle population. The number of particles is determined taking into account the length of the sequences and the number of sequences to align.
In order to test the algorithm, seven protein families of different length and different percentages of sequence identity were selected from the alignments database BALiBASE, available in [10]. One protein family was selected from each length category (short, medium and large), and one from each range of identity percentage (less that 25%, between 25 % and 40% and greater that 35%). Figure 8 presents the protein families used in the experiments.
These protein families were previously aligned using the well-known algorithm Clustal X 1,8. The alignment obtained using Clustal was evaluated using a PAM250 matrix. The alignment was obtained using a penalty of 10 for each gap. The alignment produced was used as the starting point of the PSOAlign algorithm. The algorithm stopped after 10 consecutive iterations without improving the quality of the solution.
Figure 9 shows the results of the experiments. The best results were obtained with the proteins of medium identity percentage (in general, higher than 20%). Only in a few cases the algorithm did not improve the quality of the results produced by Clustal, most of them were protein families with low identity percentage. In most cases, the proposed algorithm was able to improve the results obtained by the Clustal X algorithm.

VI. Conclusions

In this work an algorithm based on swarm intelligence, particularly the PSO algorithm, was proposed to address the multiple sequence alignment problem. The proposed approach was tested using some protein families, in order to improve the alignments generated by the clustal X 1.8 algorithm. In most of the cases, the proposed algorithm significantly improved the results obtained by clustal X.
In the proposed algorithm, every aligned sequence was taken as a coordinate, and a distance measure between alignments was proposed, as well as a movement mechanism for the particles.
It is important to notice that the results obtained depend highly on the initial set of particles. Additionally, the speed of the algorithm can be improved through a parallel implementation.
In future work, additional experimentation should be performed, including experimentation with sequences of nucleic acids, the use of other algorithms to find initial sequence alignments and the use of other scoring schemes based on PAM or BLOSUM matrices. Some heuristics for the initialization of the swarm could be also studied.

References

[1] Goldberg, David E. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Pub. Co. (1989).

[2] Mount, David E. Bioinformatics: Sequence and Genome Analysis. Cold Spring Harbor Laboratory Press. (2001).

[3] Smith, T.F. y Waterman, M.S.. Identification of Common Molecular Subsequences. J. Mol. Biol. 147. 195-197. (1981)

[4] Feng, D.F. and Dolittle, R.F. Progressive sequence alignment as a prerequisite to correct phylogenetic trees J. Mol. Evol. 25, 351?360. (1987).

[5] Notredame, Cedric and Higgins, Desmond. SAGA: sequence alignment by genetic algorithm. Nucleic Acids Research, Vol. 24, No. 8. Pags. 1515?1524. (1996)

[6] Higgins ,D.G. and Sharp,P.M. CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. Gene, No. 73, 237-244. (1988)

[7] Bonabeau, E., Dorigo, M. and Théraulaz G. From Natural to Artificial Swarm Intelligence. Oxford University Press. (1999).

[8] Kennedy, James y Eberhart, Russel. Particle Swarm Optimization. Proc. IEEE Int'l. Conf. on Neural Networks. IEEE Service Center, Piscataway, NJ, IV:1942-1948.

[9] Atwood, Teresa. Introduction to Bioinformatics. 2nd. Ed. Prentice Hall. Bogotá, Colombia. 115-119. (2002).

[10] Thompson, Julie; Plewniak, Frédéric and Poch, Olivier. BALiBASE: A benchmark alignments database for the evaluation of multiple sequence alignment programs. Bioinformatics, 15, 87-88. (1999). Available at: `www-igbmc.u-strasbg.fr/BioInfo/BAliBASE/align_index.html`