Regular Expressions Fusion using Emergent Computing

RAFAEL TORRES[1], JUNIOR ALTAMIRANDA[2], JOSE AGUILAR[3]

[1]Centro de Estudios en Microelectrónica y Sistemas Distribuidos (CEMISID)
University of Los Andes
Faculty of Engineering. Campus La Hechicera. Mérida
VENEZUELA
torresrafael@ula.ve

[2]Centro de Estudios en Microelectrónica y Sistemas Distribuidos (CEMISID)
University of Los Andes
Faculty of Engineering. Campus La Hechicera. Mérida
VENEZUELA
altamira@ula.ve

[3]Department of Computer Science, Centro de Estudios en Microelectrónica y Sistemas Distribuidos (CEMISID)
University of Los Andes
Faculty of Engineering. Campus La Hechicera. Mérida
VENEZUELA
aguilar@ula.ve

CHRISTIAN DELAMARCHE
Structure et Dynamique des Macromolecules
University of Rennes I
Campus de Beaulieu, Nb 13, Rennes
FRANCE
Christian.delamarche@univ-rennes1.fr

Abstract: - In this article is presented an approach for the comparison and establishment of patterns (motifs) from regular expressions that denote protein families, through the fusion of the sames. This task has a high computational cost, for this reason our approach uses a combinatorial optimization algorithm based on Ant Colonies to obtain the patterns that help the study and classification of families of protein chains. Particularly, we propose a system that can efficiently generate a pattern of the fusion of two regular expressions that describe protein chains, denoted in PROSITE language, in order to facilitate the analysis and classification of protein families from the structural point of view.

Key-Words: - Bioinformatics**,** Artificial Ant Colony, Combinatorial Optimization, Regulars Expressions, Motifs, Amyloid Protein.

1 Introduction

The discovery and the identification of patterns (motifs) in sequences of amino acids are part of one of the main objectives in biosequences analysis [1]. One of the tasks in this domain is the protein patterns fusion described by means of regular expressions denoted in PROSITE language [2], [3], which allows to know to how the proteins evolved and if exist relationship between them.

Currently, the algorithms that allow to find the optimal fusion of these motifs (patterns) have poor performances (at level of time and computational resources utilization) [4], [5], so it is necessary to develop new algorithms using emerging computational techniques. The solution proposed in this paper is based on a computational technique known as Ant Colony Optimization, which is based

on the behavior of ants to find the shortest path between a nest and a food source [6], [7].

2 Problem Formulation

A Motif is a region or portion of a protein sequence that possesses a specific structure and describes a specific function. Different representations of motifs have been proposed; in this paper we use a deterministic representation. Deterministic motifs are described by a regular expression. On the other hand, PROSITE is a motifs database with biological relevance that describes the motifs using a set of rules for represented them like regular expressions [2], [3]

The identification and fusion of motifs in the amino acid sequences is one of the fundamental aims in the modern biology. It's implications in the study of proteins known will allow the discovery of new sequences of amino acids with structures and functions that might help to the treatment of diseases. For this reason, it's necessary the search of new approaches to find the motifs, denoted in PROSITE language, with biological sense, in a group of proteins. Particularly, is very important to find if there are similarities between them, by then to construct a general pattern of them. The patterns found may be explained by the existence of segments that have been preserved by the natural evolution of proteins, and suggests that the obtained regions play a functional and structural role in these mechanisms.

3 Problem Solution

We propose an algorithm for the fusion of protein motifs denoted as regular expressions. This algorithm can efficiently find the union between two regular expressions described by PROSITE [2], [3], and allows the generation of a new regular expression.

This algorithm is based on the Ant Colony Optimization [6], [7], [8] with some modifications with the purpose of to fit to the fusion of protein motifs. In each execution of our algorithm, two regular expressions are fused (see Fig. 1). In general, the macro-algorithm for the fusion process of protein motifs is as follows:

1) Create the route graph.
2) Walk of the ants on the route graph.
3) Choose the best nodes
4) Build the Resultant Regular Expression.

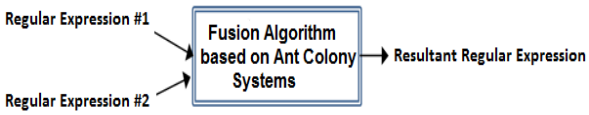

Fig.1. Fusion of two regular expressions

2.1 Create the route graph

Because the problem of fusion of protein motifs emerges from the study of the primary structure of proteins, which is a linear structure consisting of the amino acids constituent of a protein, there are two basic conditions for the design of the graph where will walk the ants:

The first stems from an analysis in the construction of motifs, which shows that is essential for this task the position of different amino acids along the protein chains, which can be viewed as one-dimensional arrays. So, a two-dimensional data structure will represent the two motifs. In the second one, we establish that the product of the fusion of motifs must generate a new pattern from which to build new amino acid chains that belong to regular expressions fused.

For the previous reasons, our graph will be represented in the plane, and each node will have arcs at the right and left sides, in this way the ants can only move them in horizontal direction. The nodes must store the pheromone level deposited by the ants that visit them and the biological information about the amino acid that represent (see Fig. 2). This information will be constituted by the type of amino acid that represents, and the family to which it belongs (see Table 1), or an identifier for special nodes (see Table 2)

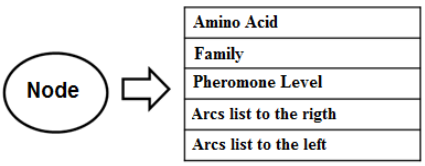

Fig 2. Data structure of a node

Amino Acids Family	Amino Acids	Classification
Aliphatic Amino Acids	G A V I L M	1
Aromatic Amino Acids	F Y W	2
Basic Amino Acids	K R H	3
Neutral Amino Acids	S T N Q	4
Acid Amino Acids	D E	5
Sulfur Amino Acids	C	6
Imino Acid	P	7

Table 1. Classification and family of the amino acids

Information	Special identifier	Classification
Gap	x	0
Empty	_	-1
Start	Start	-2
End	End	-2

Table 2. Identifiers for special nodes

ISBN: 978-960-474-258-5

For the graph construction we transform a regular expression to a stack data structure (for ex., see the regular expression S-A(1,3)-x-[KV] in Fig. 3)

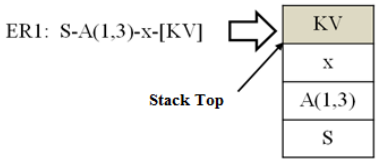

Fig. 3. Transformation of a regular expression to a stack

Additionally, two nodes are defined, that serve as guide for the construction of the graph, to indicate the beginning and end of the route (see fig. 4). Then, we proceed to extract the elements that are at the top of the stack iteratively, and built the nodes in the network (amino acids) which are in the same position in the graph. Also adds a node gap, which will serve as an auxiliary route for cases in which the ants must not continue for any of the available nodes. In this way, we avoid that an ant stops itself. On the other hand, when we extracts a gap from the stack it is not necessary any additional node.

For the special case when there are 2 values within the parentheses, we define a special node, called empty "_", to avoid the deadlock. It is necessary because the arcs that lead to these nodes must meet certain conditions: when an agent decides to go to an empty node, it would continue its route by nodes of this type until it does not find another node empty. For example, in the case of the Fig. 3 "A(1,3)" indicates that it is possible to have one to three Alanines, for this reason we need to include a given number of items as empty positions ((in this case 2).

Finally, when the stack is empty we stop the construction of the graph. In our approach, we build the route graph using the first regular expression to fuse.

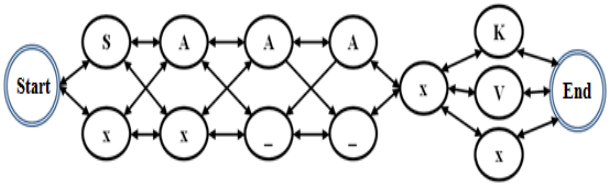

Fig 4. Route graph of the regular expression of the Fig. 3

2.2 Walk of the ants on the route graph

The artificial ants colony, like in natural ant colonies, evolves by the actions performed by its members. This way, the route graph is walked by the N-ants that constitute the colony. So, it is necessary to define the number of individuals of the colony, before they begin to walk on the route graph. On our case, each ant has a route map defined by the second regular expression to fuse. We define an ant type data structure composed of 9 elements, whose characteristics are described in Table 3. It contains the information necessary in order to that the ants can walk on the route graph. For example, this information determines whether an ant can visit a given node, and help to establish the amount of pheromone that it must deposite on each node visited.

Element	Characteristics
Start Node	Address of the node where start the ant to walk the route graph
Route Map	Stack that contains the regular expression that must follow the ant, and serves to know that nodes should be visited by the ant in the route graph
Pheromone Increase Coefficient	Real number (0,1), it is used to establish the pheromone concentration that deposits the ants in each visited node of the route graph.
Equalities Similarity Index	Integer number [0.10], it determines the pheromone level deposited by the ant, when the node found in the graph is identical to the expected to the route map.
Families Similarity Index	Integer number [0.10], it determines the pheromone level deposited by the ant, when the amino acid found in the route graph is not equal to the route map, but belongs to the same family of amino acid
Differences Similarity Index	Integer number [0.10], it determines the pheromone level deposited by the ant, when the amino acid found in the route graph is not equal to the route map, and does not belong to the amino acid family
Gaps Similarity Index	Integer number [0.10], it serves to mark the selected node, if node type is a Gap.
Approving Similarity	Integer number [0.10], that indicates the level of minimum similarity necessary to consider that the visited node is similar. If visited node has a similarity index superior to the approving similarity, the ant looks for the following amino acid in the route map, otherwise it continuous looking for the same amino acid in the route graph.
Failures Maximum Number	Integer number greater to -1. It is the number of no successful searches before to look for a new amino acid in the route map. It serves to avoid that the ant does not remain indefinitely looking for a given amino acid of the route map on the route graph. Thus, when this number of searches is reached, the ant decides with a probability of 0.5 if search in the route map a new amino acid or looking for the same amino acid in the route graph.

Table 3. Elements of the ant data structure

ISBN: 978-960-474-258-5

In the previous section we build the route graph using the first regular expression (ER1) "S-A(1.3)-x-[KV]" to fuse. We will fuse it with the regular expression (ER2) "L (2) - A (2) - Q". Using ER2 we build the route map of the ants. The stack of ER2 is shown in Fig. 5.

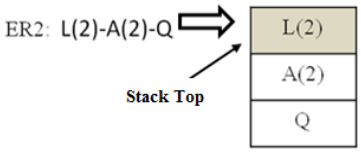

Fig. 5. Transformation of ER2 to a stack

At the beginning, the ant is placed in the initial node of the route graph, and with the route map it observes the contiguous nodes at the right side (see Fig. 6).

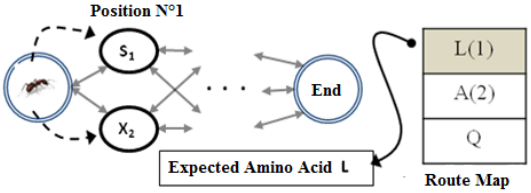

Fig. 6 Ant in the initial node of the route graph.

The ant executes the function of transition to each one of the nodes that can visit in the next position. This function consists of two phases; the first phase calculates the probability of visiting each one of the contiguous nodes ($P_n^k(r)$) based on its pheromone level 'τ_r' and the index of similarity `$\varphi_r$' of each node ('**,r**' indicates the neighboring node in the position `k', and `n' is the number of neighboring nodes at the right side for that position `k' (see Equation 1)

$$P_n^k(r) = \begin{cases} \frac{\tau_r * \varphi_r}{\sum_{i=1}^{n} \tau_i * \varphi_i} & si\ n > 1 \\ 1 & si\ n = 1 \end{cases} \quad (1)$$

The second phase decides the node to visit using the simulation of Monte Carlo. For that, we adjust all the probabilities calculated for the position `k', and choose a random number between 0 and 1. With this number we determine the node to be visited by the ant.

When the ant moves to a node deposits pheromone, that increases the pheromone concentration in the node. The quantity of pheromone deposited depend on the similarity index with respect to the amino acid waited according to the route map (Equation 2)

$$\tau_r^k = \tau_r^k + \sigma * \varphi_r^k \quad (2)$$

The similarity index is defined as follows: if the amino acid of the route graph is equal to the amino acid of the route map of the ant, then we use the equalities similarity index; otherwise, if both belong to the same family, then we use the families similarity index, otherwise, if the visited node contains gap, then the gaps index is used; otherwise, is used the differences similarity Index. In our example, the final route of an ant using ER2 is observed in the figure 7.

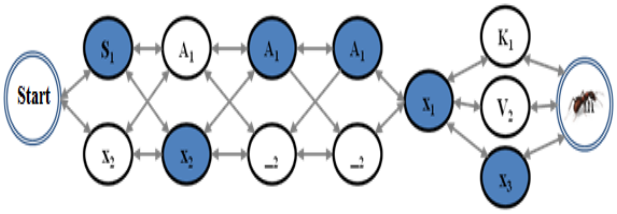

Fig. 7 The final route of the ant

For a colony, the previous process is repeated for each one of the ants of the colony. Additionally, the same process is executed recursively until the number of colony cycles desired. At the end of a cycle, there is an evaporate pheromone traces, decrementing the pheromone levels of all nodes in the graph, as shown in equation 3, where "ρ" is the pheromone evaporation coefficient.

$$\tau_r^k = (1 - \rho) * \tau_r^k \quad (3)$$

2.3 Choose the best nodes

Once the colony has completed its work, we delete the arcs that lead to those nodes with a pheromone level below the pheromone threshold that the user has defined (for our example, we fixe the pheromone threshold to 1,0), which help to preselect to the amino acids that contribute to the best solutions. Figure 8 shows the nodes selected because they exceeded the threshold (in blue).

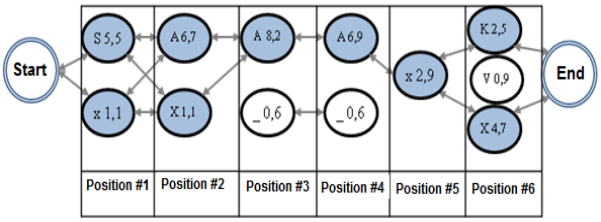

Fig. 8. Route graph with the pheromone levels of each node

2.4 Build the Resultant Regular Expression

Finally, the route graph modified is filtered to delete irrelevant information and to define the resulting patterns. To carry out this task we analyze the marked nodes of the graph, position by position, and insert the amino acids selected in a list of chains that will contain the value of the amino acids corresponding to each pattern position. To achieve this goal the following criteria are used:

ISBN: 978-960-474-258-5

1. If in the position exists only one node (amino acid) that has passed the pheromone threshold, it will be inserted in the list.
2. If exist more than one node in the same position, then we check if there is a gap or empty node. If there is not one on these nodes, we insert a string list with the amino acids for that position. Otherwise, If there is a gap or empty node, the following conditions apply:

a) If the level of pheromone of the gap node is superior to the result of the multiplication of the threshold for the rest of the nodes on this position, then we establis a new threshold for the position studied, which has the same value of the pheromone level of the gap node, and compare the rest of the nodes in the same position with this new threshold. If at least one of them exceeds the new pheromone level, then we discard the gap insertion for this position and instead adding a string with the amino acids with a higher level of pheromone that it, otherwise we insert the gap in the list.

b) If the pheromone level of the gap node is inferior, then we discard this node and a chain with the information of the remaining nodes on the same position is inserted to the list

c) We apply the same conditions of the gap nodes for the empty nodes.

We take the list that contains the amino acids corresponding to each position, and we use the PROSITE language to build the fusion regular expression (see Fig. 9)

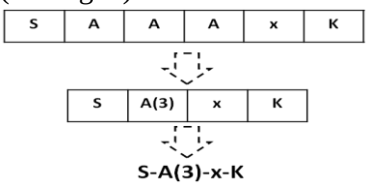

Fig 9. Resultant fusion pattern.

3 Experiments

3.1 Test

The motif proteins were taken from the database AMYPdb [9], [10]. To run the system is necessary to adjust a set of parameters. Because the number of adjustable parameters in the developed system is quite extensive, some values for the tests were left fix (see Table 4). The only parameters that we have varied are the parameters that determine the collective behavior: the cycles number and the ants number. This way, the solution depends fundamentally on the behavior of the colony.

System Parameters	Value
Pheromone Increase Coefficient	0,1
Similarity indices for the amino acids that are the same	10
Similarity indices for the amino acids that belong to a family	8
Similarity indices for the amino acids that are different	1
Similarity indices for Gaps	3
Approving Similarity Index	3
Failures Maximum number	0
Pheromone Initial level on the graph nodes	1,0
Pheromone evaporation coefficient	0,05

Table 4. Parameter List

a) **Fusion of [ST]-x(2)-[ST] with [ST]-x-[RK]**

It's possible to observe that with the expression "[ST]-x(2)-[ST]" can be obtained a chain of 4 amino acids (for example, SAKT), whereas with the expression "[ST]-x-[RK]" is obtained a chain of 3 amino acids (for example, TER). Our algorithm takes the regular expression "[ST]-x(2)-[ST]" as ER1 for the construction of the route graph (the longest), and the other regular expression is ER2 (with it the ants define the route map). In addition, the pheromone threshold is equal to the pheromone initial level in the nodes (1,0).

For the tests, thirty fusions of the two regular expressions with the same group of parameters were studied. Also, we study the average time used by the algorithm for obtaining the respective solutions and thus determine when a set of parameters is better than another.

The first tests were carried out for the parameters set: ants numbers = 4, cycles number = 4. We observe that the algorithm converges in 90% of the times (see Table 5), with an average time of 0.76 seconds.

Pattern	Pattern	Pattern	Pattern
S-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)
[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)
[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)
[ST]-x(3)	[ST]-x(3)	T-x(3)	[ST]-x(3)
[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)
[ST]-x(3)	T-x(3)	[ST]-x(3)	[ST]-x(3)

Table 5. Results for the first tests

The second tests were carried out for the parameters set: ants number = 4, cycles number = 8. We observe that the algorithm converges in a 92.33% of the times (see Table 6). In addition, the algorithm had an average time of 1.10 seconds.

ISBN: 978-960-474-258-5

Pattern	Pattern	Pattern	Pattern	Pattern
[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)
[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	S-x(3)
[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)
[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)
[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)
[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	T-x(3)	[ST]-x(3)

Table 6. Results for the second tests

The third tests are for the parameters set: ants number = 8, cycles number = 4. We see in the Table 7 that the algorithm converges in 96.66% of the times and it presents a best runtime with respect to the previous tests, the average time is 0.89 seconds

Pattern	Pattern	Pattern	Pattern	Pattern
[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)
[ST]-x(3)	S-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)
[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)
[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)
[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)
[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)

Table 7. Results for the third tests

The last tests is for the next parameters: ants number = 8, cycles number = 8, obtaining with this set a better precision level, since it has a convergence of 100% compared to the expected pattern (see Table 8). Additionally, the average time was 1.09 seconds.

Pattern	Pattern	Pattern	Pattern	Pattern
[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)
[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)
[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)
[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)
[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)
[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)	[ST]-x(3)

Table 8. Results of the last tests

The best resultant fusion pattern of the two regular expressions is “[ST]-x(3)”. (see Fig. 10). We conclude that we can obtain the fusion of the expressions [ST]-x (2)-[ST] and [ST]-x-[RK] in a very short time, when the ants and cycles number are equal to the positions number of ER1; Nevertheless, the best performance is obtained when the ants and cycles number duplicates the positions number of ER1 (the selected pattern to construct the route graph) with a very similar runtime.

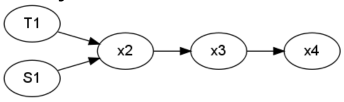

Fig. 10. Graph of the fusion pattern [ST]-x(3)

b) **Biological Patterns Fusion**

Now perform the fusion of two patterns and analyze its biological sense, to see if the patterns generated by the system are useful for the study of protein chains. The first pattern is (see Fig. 10 and Table 9):
K-x-G-S-L-[DGK]-N-[AIV]-T-H-V-[AP]-G-G-G-[AHN]-[KV]-[KQ]-I-E-[NST]-[HR]-K-L-[DST]-F-[RS]-x-[AN]-[AS]-[KP]-x-[KV]-[GT]-[DS]-[HK]-[GT]-[AN]-[EY]-[IQ]-[PV]-x-K-S-[DP]-[GV]-[HKV]

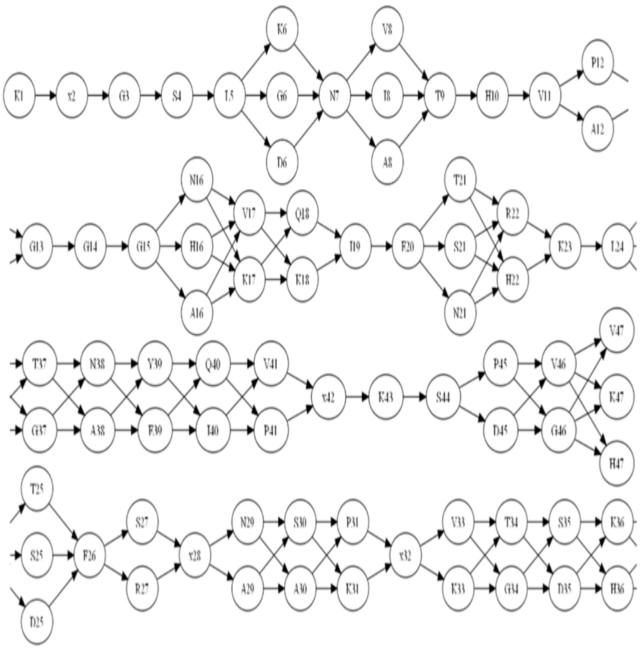

Fig 10. Graphic representation of the first pattern

Family	Protein Name	Protein description	Organism
Tau	O02592_CAEEL (O02592)	PTL-1A protein (Protein with tau-like repeats protein 1, isoform a)	Caenorhabditis elegans
Tau	Q17364_CAEEL (Q17364)	TAU-1a (Fragment)	Caenorhabditis elegans
Tau	Q17365_CAEEL (Q17365)	TAU-1b (Fragment)	Caenorhabditis elegans
Tau	Q53YB1_HUMAN (Q53YB1)	Microtubule-associated protein tau (Microtubule-associated protein tau, isoform 4)	Homo sapiens (Human)
Tau	Q547J4_MOUSE (Q547J4)	Microtubule binding protein tau	Mus musculus (Mouse)
Tau	Q5CZI7_HUMAN (Q5CZI7)	Microtubule-associated protein tau	Homo sapiens (Human)
Tau	TAU_MOUSE (P10637)	Microtubule-associated protein tau (Neurofibrillary tangle protein) (Paired helical filament-tau) (PHF-tau)	Mus musculus (Mouse)
Tau	TAU_PANTR (Q5YCW1)	Microtubule-associated protein tau	Pan troglodytes (Chimpanzee)

ISBN: 978-960-474-258-5

Tau	TAU_PAPHA (Q9MYX8)	Microtubule-associated protein tau (Neurofibrillary tangle protein) (Paired helical filament-tau) (PHF-tau)	Papio hamadryas (Hamadryas baboon)
Tau	TAU_PONPY (Q5S6V2)	Microtubule-associated protein tau	Pongo pygmaeus (Orangutan)
Tau	TAU_RAT (P19332)	Microtubule-associated protein tau (Neurofibrillary tangle protein) (Paired helical filament-tau) (PHF-tau)	Rattus norvegicus (Rat)
Tau	TAU_SPECI (Q6TS35)	Microtubule-associated protein tau	Spermophilus citellus (European suslik) (Citellus citellus)

Table 9. Some proteins chains that contain the first pattern [9].

The second pattern is (see Fig. 11 and Table 10): G-S-[KT]-D-N-[IM]-[KNR]-H-x-P-G-G-G-[KNS]-V-Q-I-[FV]-[DHY]-[EK]

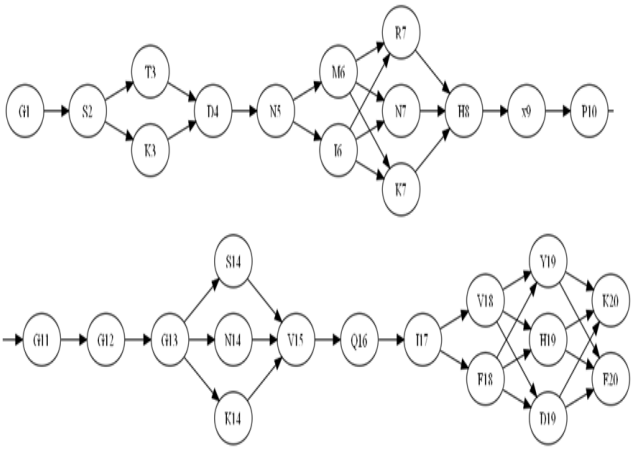

Fig 11. Graphic representation of the second pattern

Family	Protein Name	Protein Description	Organism
Tau	O02592_CAEEL (O02592)	PTL-1A protein (Protein with tau-like repeats protein 1, isoform a)	Caenorhabditis elegans
Tau	Q17364_CAEEL (Q17364)	TAU-1a (Fragment)	Caenorhabditis elegans
Tau	Q8JIW8_XENLA (Q8JIW8)	Tau-like protein-1	Xenopus laevis (African clawed frog)
Tau	Q91WK4_MOUSE	Microtubule-associated protein	Mus musculus (Mouse)
	(Q91WK4)	tau	
Tau	TAU_BOVIN (P29172)	Microtubule-associated protein tau (Neurofibrillary tangle protein) (Paired helical filament-tau) (PHF-tau)	Bos taurus (Bovine)
Tau	TAU_CAPHI (O02828)	Microtubule-associated protein tau (Neurofibrillary tangle protein) (Paired helical filament-tau) (PHF-tau)	Capra hircus (Goat)
Tau	TAU_GORGO (Q5YCW0)	Microtubule-associated protein tau	Gorilla gorilla gorilla (Lowland gorilla)
Tau	TAU_HUMAN (P10636)	Microtubule-associated protein tau (Neurofibrillary tangle protein) (Paired helical filament-tau) (PHF-tau)	Homo sapiens (Human)
Tau	TAU_HYLLA (Q5YCV9)	Microtubule-associated protein tau	Hylobates lar (Common gibbon)
Tau	TAU_PONPY (Q5S6V2)	Microtubule-associated protein tau	Pongo pygmaeus (Orangutan)
Tau	TAU_SPECI (Q6TS35)	Microtubule-associated protein tau	Spermophilus citellus (European suslik) (Citellus citellus)

Table 10. Some proteins chains that contain the second pattern [9].

To realize the fusion the parameters are taken from the Table 4. In addition, 94 ants and equal number of colony cycles used. The system execution is realised 3 times and the following results are obtained (see Table 11):

Execution	Pattern
1	x(2)-G-S-x-[DGK]-N-[AIV]-T-H-x-[AP]-G(3)-[HN]-[KV]-Q-I-x(2)-[HR]-K-(24)
2	x(2)-G-S-x-[D**G**K]-N-[AIV]-T-H-x-P-G(3)-[**A**HN]-V-Q-I-x(2)-[H**R**]-K-x(24)
3	x(2)-G-S-x-[DK]-N-[AIV]-T-H-x-[**A**P]-G(3)-[**A**HN]-V-Q-I-x(2)-H-K-x(24)

Table 11. Fusion Resultant

The data base AMYPdb is consult for the results obtained and is observed that the second and third

pattern are not found. For the first pattern the result is the following (see Table 12):

Amyloid Protein: x(2)-G-S-x-[DGK]-N-[AIV]-T-H-x-[AP]-G(3)-[HN]-[KV]-Q-I-x(2)-[HR]-K-(24)

Family	Protein Name	Protein Description	Organism
Tau	O02592_CAEEL (O02592)	PTL-1A protein (Protein with tau-like repeats protein 1, isoform a)	Caenorhabditis elegans
Tau	Q17364_CAEEL (Q17364)	TAU-1a (Fragment)	Caenorhabditis elegans
Tau	Q17364_CAEEL (Q17364)	TAU-1b (Fragment)	Caenorhabditis elegans

Table 11. Protein that contain the Pattern Resultant

4 Conclusion

The algorithm based on Ant Colony Optimization is a good solution for the discovery of protein patterns. It's possible to find fusion patterns for groups of regular expressions with biological sense. The main parameters of our algorithm are that determine the collective behavior of the ants.

References:

[1] Srinivas V., *Bioinformatics A modern Approach*, Eastern Economy Edition, 2005

[2] *Database PROSITE*. Available in: [http://www.expasy.ch/prosite/]

[3] Bairoch A., Bucher P., Hofmann K., *The PROSITE database, its status in 1997* Nucleic Acids Research, Vol. 25, No. 1, pp. 217-221, 1997

[4] Pevsner J., *Bioinformatics and Functional Genomic*s, Second Edition, Wiley – Backwell, 2009

[5] Mathura V., Kangueane P. *Bioinformatic A Concept-Based Introduction*, Springer, 2009

[6] Dorigo M., Di Caro G., Sampels, M. *Ant Algorithms*. Bruselas, Springer, 2002

[7] Dorigo M., Thomas, S. *Ant Colony Optimization.* Massachusetts Institute of Technology (MIT), Boston, 2004.

[8] Aguilar J., Rivas F. (Ed.), *Introducción a la Computación Inteligente*, MERITEC, Venezuela. 2001.

[9] *AMYPdb*. Available in: [http://amypdb.univ-rennes1.fr]

[10] Pawlicki S., Le Béchec A., Delamarche C., *AMYPdb: A database dedicated to amyloid precursor proteins*, BMC Bioinformatics, Vol. 9, pp. 273-28, 2008