Parallelization of bioinformatics methods using a computational grid

Article by Mauricio Adami Mariani[1], Samuel Brando Oldra[2], Precious Adewopo[3]
[1]*Msc, Texila American University, Brazil*
Email: mauricio.a.mariani@gmail.com
[2]*Computer Science and Information Technology Department*
University of Caxias do Sul, Brazil, South America
Email: samuel.oldra@gmail.com
[3]*Computer Science Department - School of Information Technology*
Texila American University - Guyana, South America
Email: preciousadewopon@yahoo.com

Abstract

Recent advances in computer science improve research in several areas. These include an area with great emphasis, the biology, which is the field of study of this work. The use of computer for analysis of data, and processing methods of biology is called bioinformatics. However, when are analyzed DNA sequences from complex organisms which have thousands nucleotides is required a greater processing power. Due to the increase the amount of data to be processed we can use HPCA. Within this context in this work, we study the use of computing capabilities in performance processing DNA sequences. More specifically the parallelization of methods used by the research group in bioinformatics at UCS(University of Caxias do Sul). For development this work we chose to use computational grids, since this type of platform provides a high processing capacity at low cost.

Keywords: *Parallel Computing, Bioinformatics, Computing Grid, Neural Network, DNA*

1. Introduction

Biology is the branch of science that studies the characteristics and behavior of organisms, and how these interact with each other's and with their environment. These characteristics and behaviors are defined by DNA(Deoxyribonucleic acid). Thus, the study of DNA can provide information on the genotype(data gene of an organism)[5][21]. Initially, the study of DNA was performed manually, later it moved to use computers for this task. However, with amount data increase, and the need for immediate results, there was necessary to use High Performance Computing Architectures(HPCA)[6].

As a first alternative of a high performance environment there is the supercomputer which can be characterized mainly by multiple units of processors and memory, connected via a proprietary network. But the cost is high and the acquisition for this technology is restricted[1]. Another option for achieving a high computational power is the computers clusters, which consist of a parallel environment comprising personal computers, connected by a local high-performance network. Even they cost is less than a supercomputer, they have a relatively high cost to invest in the computers, network and need maintenance[26]. An alternative to supercomputers and clusters are Grids. They are based on the use of non-dedicated computers to developing a high performance platform. Computational grids are designed to use of idle cycles of computers for processing. Since various institutions such as the UCS, have a large number of computers remain idle at various times the day this platform becomes attractive because it allows obtaining a high computational power through with low investment in equipment[27].

Often used as input data(DNA sequences) for these methods are obtained online repositories without restricted access, such as GenBank[9][21]. These databases are used, whether in obtaining data or updates for hundreds of institutions. Thus, because the large amount of data in those repositories, for processing such data is necessary use high performance computing. After the initial study of methods, an analysis was performed to adapt them to a grid platform with the aim to reduce the execution time of these tasks.

2. Molecular biology and bioinformatics

This section aims to describe the central dogma of biology. This will be addressed the basic and important concepts on this topic in order to make them understandable goals and the importance of computational methods used in the area of bioinformatics.

2.1. Molecule of DNA

The characteristics and the continuity of the bodies depend on the genetic material present cells. DNA is a molecule comprised of two tapes formed by repetitive units, called nucleotides, linked together, as seen in Figure 1. There are four types of nucleotides are denoted by the letters A(adenine), T(thymine), C(cytosine), G(Guanine) as illustrated in Figure 1[23]. In one molecule of the two strands of DNA nucleotides there are connections so that the nucleotide A in a tape binds to the T nucleotide and C nucleotide binds to the nucleotide G as can be seen in Figure 1[23].

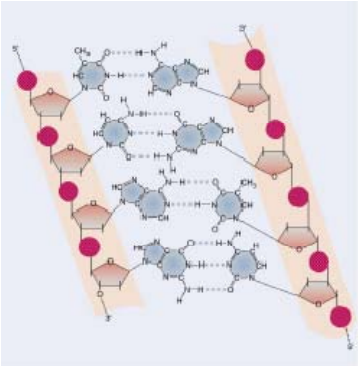

Figure 1: The mRNA takes the information from the DNA molecule in the nucleus of cytoplasm where translation occurs. In Figure 1 can see links between AT nucleotides with two bridges hydrogen's and GC with three bridges hydrogen's. The sequence is TA, CG, AT, GC(Lewin, 2001).

2.2. Central bology dogma

The central dogma defines the paradigm of molecular biology and includes three processes: replication, transcription and translation of DNA. DNA replication is a process in which a cell divides to form other, and this process is a semi-conservative type, since each strand contains a double helix generated old and other newly synthesized filament[20]. This process is responsible for the passage of information one generation to another. The transcription process converts the information into a DNA strand of RNA(mRNA), which will serve for translation. Since the translation process converts the information contained in the mRNA into proteins [23]. Transcription is the process by which a strand of DNA, called tape mold form single-chain mRNA by replacing the nucleotides A by U (Uracil), T by A, C and G by C to G (Lewin, 2001). Finally the translation is when every three nucleotides in that molecule are denominated mRNA codon and corresponding an amino acid and the set of amino acids will be translated in a protein[7](Lewin, 2001).

2.3. Methods used

The sequential implementation of the follow methods studied: calculation of Entropy, calculation of Periodicity, calculating the Clustering Coefficient and Calculation of the Dispersion Coefficient Clustering were developed by Prof. PhD. Gunther Johannes Gerhardt Lewczuk using the language C. The last method studied, Determination of Promoters through a Neural Network was developed and implemented by the Prof. PhD. Scheila de Avila e Silva using the R language[15].

2.3.1 Calculation of entropy

The Entropy is a thermodynamic magnitude usually associated with the degree of disorder of a the measuring system and of the energy can't be transformed into work. This is also natural phenomena to the following extent: complexity, compressibility, and predictability Randomness[28]. One of the main reasons for calculating the entropy be used in DNA molecules lies in the fact that with the its uses can

obtain the degree of organization of a sequence[22].

Figure 2: Sequence fully organized , with entropy 0 because there are only the nucleotides a

In Figure 2 there is a DNA sequence fully organized, i.e. a sequence whose entropy is 0. When the calculated entropy of a message is computed, for example, probably we will know the structure behind this message. The reason to apply the method of entropy in DNA sequences is the search for patterns. Through these patterns can searching regions of DNA that have specific meaning, such as the boundaries of the coding areas DNA, or even get some standard, and from, this check for some meaning organic[28].

For the calculation of the entropy is used as input a DNA sequence called window. This sequence S is composed of k nucleotides, where each element $x_i \in \{a, t, c, g\}$ and may be written as $S = \{x_1, x_2, \ldots, x_k\}$. Initially this method considers the amount of existing equal triplets DNA sequence, beginning at x_1, x_2, x_3 and moving one position to the right, so second triplet is x_2, x_3, x_4 and so on. Thus one obtains a set $(S) = \{j_1, j_2, ..., j_n\}$. With n positions, comprising $j_1 = Q(x_1, x_2, j_2 = x_3)$, $Q(x_2, x_3, x_4)$, ..., $j_k = Q(x_{k-2}, x_{k-1}, x_k)$, where $Q()$ indicates the number of times each sequence (single) triplet appears in the window.

This set can have a maximum of 64 positions, or 4 possible nucleotides combined in three positions. For the set $D(S)$, one can construct a discrete probability distribution $P = (p_1, p_2, ..., p_n)$ being p_i the probability of each sequence of triplets and which $\sum_i p_i = 1$, resulting in 1 because it is the sum of the probabilities p_i of each triplet is $p_i = \frac{j_i}{k}$. After obtain the distribution P can calculate the Shannon entropy of a discrete probability distribution by the formula[29]:

$$H(S) = H(D(S), P) = -\sum_{i-1}^{n} p_i log_2 p_i$$

2.3.2 Periodicity Search

This method seeks to establish relationships between the various biological patterns existing along a repetition sequence. The search periodicity occurs through the use of correlation functions (CF) that perform the count of the regions of nucleotides in a given range. For example, if the periodicity is 3, a comparison of a nucleotide with its two neighbors underlying will be made, as shown in Figure 3.

Figure 3: Distance of two nucleotides 3 characterizes the periodicity

On this CF various tools which can locate or map these regions, such as the Fourier transform, that maps an area in the CF frequency can be used. The periodicity 3 is the most studied since it is indicative of coding regions[16].

The search method uses frequencies up of an initial window of size k, where it is applied to the Kronecker correlation function for comparing each character in the window studied. The algorithm is the sequence of nucleotides as a circular chain[17]. In correlation function Kronecker each nucleotide is compared to all other window. The comparison returns 1 if two elements are equal and 0 if they are different.

Thus, $S(1)$ is the sum resulting from the comparison between the distance element 1, $S(2)$ is the sum of the comparison between the distance elements 2 and so on. Then $S(k)$ is the value of the correlation function to a distance between elements of the sequence k. This function is represented mathematically by

$$S(k) = \frac{\sum_{i=F}^{L} \delta_{i,((i+k+F) mod (L-F+1))+F}}{(L-F+1)}$$

where F and L represent the first and last elements of the window, respectively. The mod operator returns the remainder of division between the terms i + k + F and F + L 1, and d is the Kronecker delta function defined by:

$$\delta_{i,j} = \begin{cases} 1 \text{ for } x \neq y \\ 0 \text{ for } x = y \end{cases}$$

where *i* and *j* represent the positions of *x* and *y* represent the basis elements contained in these positions. Thus, in this correlation function are contained in the bases and *j* are equal, the Kronecker delta function returns 1, and otherwise returns 0. Using this correlation function can obtain the relationship between each element in the window relation to all other elements in the same window.

Is then removed the DC (Direct Current) from the result of the correlation function. This is done by subtracting the mean of each outcome the same for both, and need to add up all the results of the correlation function and divide by the size of the selected window. This process is also known as Zero Frequency therefore aligns the result of the correlation function in respect to $y = 0$, to facilitate data analysis.

About result of removing DC is applied to Fourier transform. From this you can see the correlation of each nucleotide by means of a frequency plot. This transformation of values in frequency space is given by the equation:

$$F\{S(n)\}(w) = \int_{-\infty}^{\infty} \frac{S(n)e^{-2\Pi win}dn}{\sqrt{2\Pi}}$$

where *i* is the imaginary unit $\sqrt{-1}$ and e Euler number. Thus with the application of the Fourier transform creates a space frequency due to the displacement S_k, where $F(w)$ is given by the signal $S(k)$ in the frequency domain, i.e. the spectrum.

2.3.3 Calculus of clustering coefficient

Using a graph theory to analyze the clustering coefficient sequence is another method used in this work. Initially, the DNA sequence is transformed into a graph[13], where each graph vertex corresponds to a triplet of nucleotides, and two vertices are connected by an edge if its nucleotides are juxtaposed[4], as can be seen in Figure 4.

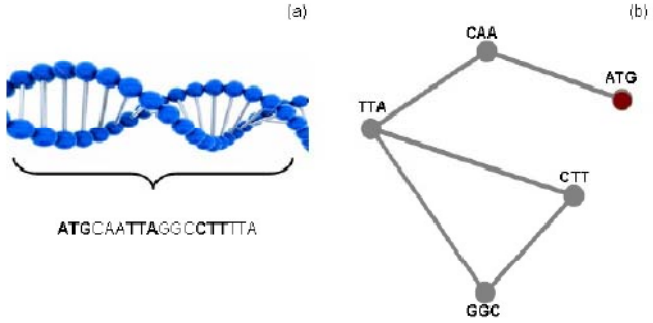

Figure 4: (a) Sequence of nucleotides. (b) Graph of the sequence (a). Each vertex is a triplet of nucleotides starting at ATG.

Then, in the resulting graph clustering method that checks the density of triangles in order to determine whether this graph is arranged is used. This measure lets you check whether or not the graphs have a standard in its structure. If the pattern does not exist, their clustering coefficients is similar to a random graph; But when you have a standard structure, you can find evolutionary traits[13].

This method has as input a sequence of nucleotides S. Initially this method transforms this sequence S in an undirected graph where the vertices are triplets of nucleotides and a link is established between two triplets, when these are juxtaposed in the DNA sequence. The method uses the first two triplets that are juxtaposed and assigns each triplet a single value, forming an edge between these two vertices (points), and so the triplets juxtaposed by the end of the sequence S. For the representation of the graph used a structure of m adjacency matrix of order N, where the element m_i is 1 if node *i* is connected to the vertex *j* else if 0 if not connected.

The clustering method aims to calculate the level of clustering of vertices. c_i representing the clustering coefficient for this level of grouping is a value given by a vertex *i*, that is, calculate the number of edges

between vertices adjacent to vertex *i*. If vertex *i* is connected to a subgraph *C*, the number of links between the vertices of this subgraph *C* is calculated by:

$$L_i = \sum_{j=1}^{L} \left(m_{i,j} \left[\sum_{k \in C} m_{i,k} \right] \right)$$

To find c_i we must normilize L_i for all possibles links k_i between the vertex *i* and the vertex of subgraph *C*, throwout:

$$c_i = \frac{2L_i}{k_i(k_i - 1)}$$

When $k_i = 0$ or $k_i = 1$ we can define $c_i = 0$. Finally *c*, which corresponds at the clustering coefficient for the graph, is obtained by averaging the coefficients of clustering of all sub graphs , as follows:

$$C = \frac{1}{L} \sum_{i=1}^{L} c_i$$

2.3.4 Calculation of the dispersion coefficient clustering

One can also apply the calculation of the dispersion over a clustering coefficient. The calculation of the dispersion is a statistical measure used to determine the location of a DNA sequence in relation to another generated as a control group[4]. Initially must be taken to control network, which often consists of a random graph. This network of control varies along its GC content of the sequence to contain only G and C nucleotides[4].

Applying the calculations of the dispersion, one obtains a graph of how the actual DNA sequence resembles a random graph. Figure 5 is a graph which illustrates this method. In the graph, the continuous line represents the random string and the points represent the actual DNA sequence. The calculation of the scattering coefficient can be important for determining the evolution of an organism DNA[4].

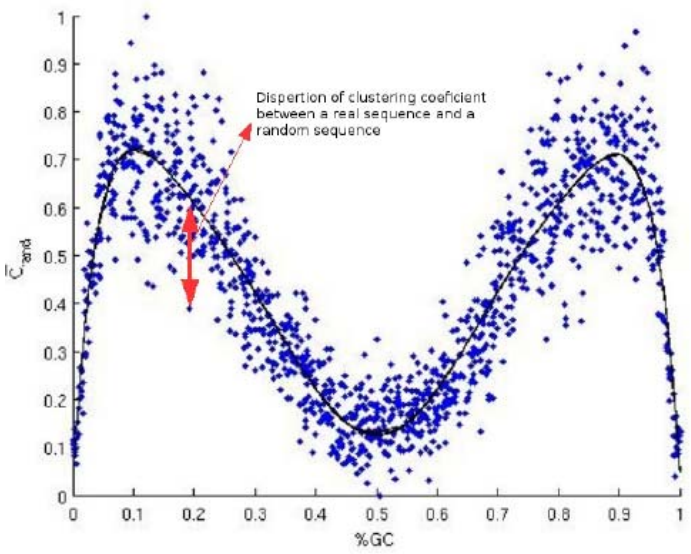

Figure 5: The black curve represents a random sequence of DNA, and the points belong to the actual DNA sequences.

The method of calculation of clustering coefficients is used as an input parameter for calculating the dispersion. From this information, the method calculates the dispersion coefficient D by the sum of the difference between the clustering coefficient of a real graph and the clustering coefficient of a random graph, divided by the standard deviation. This sum is the scattering coefficient of the graph, or the nucleotide sequence with respect to a control sequence. The dispersion coefficient D, used in this work, is defined as[16]:

$$D(C)_L = \frac{1}{n_b} \sum_{i=1}^{n_b} \frac{C_i - C_{rand}}{\sigma_{rand}}$$

where n_b is the number of windows of size L of the sequence analyzed, C_{rand} is the clustering coefficient of the control group and C_i is the clustering coefficient of the graph. In

this context, the standard deviation is C_{rand}, which can be found numerically.

2.3.5 Determination of promoters through a neural network

For certain gene has a gene expression performed correctly, a previous sequence it should be recognized. This sequence is called the promoter, since it promotes or inhibits the transcription of a gene subsequent to it (Lewin, 2001).

Recognize and predict these sequences, difficult to experimentally identified, implies knowing the network of metabolic regulation of a particular organism. Machine learning(ML) techniques are employed to recognize the promoters, among which stand out, artificial neural networks[8]. A Neural Network (NN) is a system inspired by the ML functioning of biological neural networks. NN is considered a distributed parallel processor consisting of simple processing units interconnected or not. These units are called neurons and has a natural propensity for storing experimental knowledge and making it available for use[14].

Can affirm that the RNs learn from examples. An NN is characterized: by the pattern of connections between neurons (called architecture); the method of determining weights in the connections (called training); and its activation function[25]. In the group Bioinformatics UCS uses to NN for prediction and recognition of promoter sequences in bacteria. Neurons(Figure 6) are connected by directed links, each link has a numeric weight W_{ij} associated with it, which determines the intensity and the sign of connection. In addition, each bond has an activation a_j [18]. The function of the input is given by:

$$en_i = \sum_{j=0}^{n} W_{ji} a_j$$

Then, as shown in Figure 7, an activation function that is applied a summation to derive the output:

$$a_i = g(en_i) = g(\sum_{j=0}^{n} W_{ji} a_j)$$

Importantly, there is the inclusion of external parameter of artificial neuron, a $W_{0,i}$, connected to a fixed bias input $a_0 = 1$. The term $W_{0,i}$ set the actual limit for united direction $\sum_{j=0}^{n} W_{ji} a_j$ that exceeds the unit is activated when the weighted sum of "real" $W_{0,i}$ [18] entries.

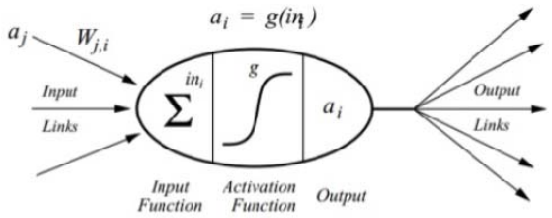

Figure 6: Model of an artificial neuron. The activation of the output unit is a_i and where a_j is the activation of output unit j and W_{ji} is the weight on the link from unit j to that unit[18].

The architecture of the NN used in this implementation is the Multilayer Perceptron. This is characterized by having input neurons, hidden neurons and output neurons. The function of the hidden neurons is to intervene between input and output efficiently. The advantages of adding these is hidden layers increases the number of hypotheses can represent the NN and thus, it is possible to extract higher order statistics. However, when you have an RN with many hidden layers, these are computationally less efficient, since they require a longer processing time[18].

To verify the accuracy of an NN, is used herein, the cross-validation technique or K-fold cross-validation (k-FCV) which is to randomly divide the pattern file into k parts demesmo size. Thus, the generation of files for training and validation occurs. Initially, the promoter sequence represented by nucleotides is divided into 10 equal parts after is translated into numbers, and for each letter is assigned a binary sequence. At the end of the sequence is placed the value 1 indicating that it is a promoter sequence. This process is also performed for a randomly generated sequence, in which case the value 0 is assigned is

not indicating that promoter. These values are used for training and testing of newborns.

After the part k_{th} dividing the promoter sequence is joined to the k_{th} part of the division of generating random sequence input from the k_{th} trial, and so on for all k files tests. As for the training inputs are concatenated all parts minus the k_{th} part of the promoter sequence and all under the k_{th} parts of the random sequence generating the file k_{th} training. This process is performed for each training file. The training and validation steps are repeated k times, being used for training $k - 1$ files and to validate the k_{th} file is not used in training. With each interaction, the validation file has a different k[18].

The computational cost required by these studies is high. Thus, these methods may contribute parallelization to decrease processing time. The use of a HPCA aims to share these problems so that can be run on different computers simultaneously. The main concepts of distributed computing and parallel computing will be presented in the next section.

3. Parallel computing

The bioinformatics tools presented in this article, require high processing capacity mainly due to large amount of data and performing complex calculations. Thus the implementation of these tools on a single computer is impractical, requiring the use of an alternative technology. Among the alternatives, the most appropriate is the parallel computing. Parallel computing aims to split a task into smaller parts and their distribution among different processors working simultaneously, or in parallel. However, these units are synchronized and communicate[12].

Parallel computers can be divided into: architectures with shared memory or distributed memory architectures. The difference between the two is that while the shared memory architecture processing units use a memory space with a single address, the distributed memory architecture each unit has its memory and these are connected through a communication network, as illustrates the image below[24].

3.1. Architecture with Shared Memory

A shared memory architecture refers to the fact that shared memory is a single globally for all processing units[12]. Thus, there is a common memory space used to exchange information between processors, for example multiprocessors. One of the advantages of using shared memory is the direct access to data in memory using only the bus to the traffic information. A disadvantage referred the dispute for access to memory, because all processors may require access to memory simultaneously, which makes the bus a bottleneck. Another disadvantage relates to the expansion system, since with the increase in the number of processors the system can become slow due to increased contention for the bus.

3.2. Architecture with Distributed Memory

On architectures with distributed memory also known as multicomputer, the processing units are independent and each of them has its own memory. These units communicate through a messaging mechanism for a computer network. Therefore they are also called messaging systems[12]. An advantage of this architecture is the easy scalability of the system, because it can increase the capacity of processing just adding computers. Among the main disadvantages are the need to access information from another computer via a network, where latency is high and the bandwidth is low compared to a shared memory environment[12]. The best known examples of architectures with distributed memory are clusters and computational grids.

3.3. Computational Grids

We chose to use this architecture for the tasks to be processed by the methods described in Section 2 are of the bag-of-tasks, i.e. not require communication between them. Thus the computational grids become a good alternative since they are in a high performance platform with low investment. To develop this work we used the GridUCS, which is a computational grid formed by approximately 300 of UCS computers labs. In the following sections we will discuss the main tool used for the management in GridUCS, the OurGrid, and use of this type of platform.

3.3.1 OurGrid

OurGrid is a computational grid of open source tool developed by the Federal University of Campina Grande (UFCG) in partnership with Hewlett-Packard(HP). This is characterized as the use of peer-to-

peer(P2P) technology, and its main purpose is to use the idle time of machines not dedicated to the processing of Bag-of-Tasks tasks, i.e., tasks that are independent and require no communication between them[10][11].

In OurGrid all processing resources belong to OurGrid network, and can be accessed by any user of it[19]. This makes use of Favors policies for the distribution of resources, where the more resources a user to assign more resources it receives network in exchange for those who have used the network. If a user does not do so automatically OurGrid, and gradually withdraw this user priorities, being indebted to the community[19]. OurGrid was the tool used in this work because it is the tool that is used in the computational grid of UCS(GridUCS). It was decided to the use of due OurGrid mainly by simplicity of configuration, management, and use of the same, compared with other management tools computational grids.

4. Implementations and Development

These implementations were adaptations of sequential algorithms of the methods described in Section 2, in order to run in the grid. The entropy calculation methods, frequency of calculation and clustering coefficient have as input a DNA sequence in FASTA format[9] that can be obtained from public repositories. The results generated can be analyzed by experts, or may be used as input for other methods. For example, the output of the clustering coefficient calculation method is used as input to the dispersion method of calculating the clustering coefficient. Already the method of determining promoters through RN has input promoter sequences, which can be obtained by RegulonDB repository[3] and has as output statistical data for further analysis specialists.

4.1. Calculation of entropy, frequency and clustering coefficient

The methods entropy calculation, frequency calculation and calculation of clustering coefficient have the same execution flow for a Grid. Their respective implementations for execution on a computing grid were also similar, and so will be detailed in this section. These methods have as input files in Fast format[9], which generally have a large number of base pairs. Also, often, these methods are applied to multiple input files. Simplified grid operation way is to assign each machine a task.

In this case it can be done by dividing the Fast files into smaller files, or files that contain only part of the main file. After splitting these parts are processed in each grid machine. For the division of the input files was developed a script in Python language. In the list below we has an excerpt of this script. Initially this script lists all Fast files from current directory and then the names of these files are stored in a list (line 1 and line 2). After the list is traversed and the files are divided into smaller parts, where each part is identified with the name of the original and a sequential number to the end. This division is based on the number of rows that each partial file should contain, and this value is a parameter reported to the script.

```
1  lista_diretorio=commands . getoutput( "ls" )
2  lista_separada= lista_diretorio . split( '\n' )
3  n_lin = sys . argv [ 1 ]
4  for  arq_fasta  in lista_separada :
5          if ( arq_fasta [−5:]== ' fasta ' ):
```

After this division is generated the file to be executed on the grid. For the generation of this file was developed another script in Python language. In follow list, there is a stretch of that script. This lists all files resulting from division performed previously (line 2), and each of these files is added to the grid submission file.

```
1  pedacos_fasta=open ( arq_s_ext+ ' . txt ' , 'w+' )
2  for i in range ( 0 , gpos +1) :
3          script_grid . write ( ' task :\n ' )
4          script_grid . write ( '\t '+ ' init :\ t '+ ' put '+ arq _ s _ ext + ' _ '+ str ( i ) + ' . fasta '
                  +      arq _ s _ ext + ' _ '+ str ( i ) + ' . fasta \n ' )
5          script_grid . write ( '\t\t '+ ' put S_dna S_dna\n ' )
6          script_grid . write ( '\t\t '+ ' put script_grid . sh script_grid . sh\n ' )
7          script_grid . write ( '\t '+ ' remote : . / script_grid . sh ' + ' ' + arq _ s _ ext + ' _ '
```

```
                + str ( i ) + '\n ' )
8           script_grid . write ( '\t '+ ' final : get '+arq_s_ext+ ' _ '+ str ( i )+ ' .tar.gz '+
                arq_s_ext+ ' _ '+ str ( i )+ ' . tar . gz\n ' )
9           pedacos _ fasta . write ( arq _ s _ ext+ ' _ '+ str ( i )+ '\n ' )
10  pedacos _ fasta . close ( )
```

In follow list we give an example of a task submission file. The beginning of the file is indicated with the reserved word job, and the name of the run. This name is assigned by the reserved word label (line 2). The snippet of a task to be performed on the grid is initiated by the reserved words task and init (line 3 and line 4). The put command corresponds to the files to be sent to the grid of computers (lines 4, 5 and 6), and the first attribute of the put command match the file name to be sent and the second the name that will receive this file on the remote machine (grid computer). Have the remote command corresponds to the command to be executed in the grid computers (line 7).

Finally, with the final and get commands, files are transferred from the grid machine for submission machine (line 8). At the command get the first attribute matches the file name to be searched in the grid machine, and the second attribute is the name that the file will receive the return.

```
1   job :
2           label : Mauricio
3   task :
4           init : put Kineococcus_radiotolerans_SRS30216_0 . fasta
5   Kineococcus_radiotolerans_SRS30216_0 . fasta
6                   put S_dna S_dna
7                   put script _ grid . sh script _ grid . sh
8           remote :. / script _ grid . sh 4096 Kineococcus_radiotolerans_SRS30216_0
9           final : get Kineococcus_radiotolerans_SRS30216_0 . s
                Kineococcus_radiotolerans_SRS30216_0 . s
```

After the execution of the tasks on the grid another script, also implemented in the Python language, is responsible for uniting all results files. A piece of code responsible for this task is given in list below. Initially you create a directory where the files are stored with the results (line 1). Following are listed all result files that have the name of the original file Fast (line 2). These files are opened and the contents of these is stored in a list where each position of this list corresponds to a row of result files (line 3 line 6). After this result file is removed (line 7). Finally, the list of results is written to an output file, and thus concatenating all result files (lines 8 and 9). At the end of this file is moved to the directory you created to store the results (line 10).

```
1   res=commands . getoutput ( " mkdir resultado " )
2   for i in range ( 0 , n_arq ) :
3           arquivo=open ( arq _ orig+ ' _ '+ str ( i )+ ' _ '+ str ( size )+ ' . s ' , ' r ' )
4           linhas=arquivo . readlines ( )
5           linhas =[ line . strip ( ) for line in linhas ]
6           arquivo . close ( )
7           res=commands . getoutput ( "rm "+arq _ orig+ ' _ '+ str ( i )+ ' _ '+ str ( size )+ ' . s ' )
8           for l in linhas :
9                   saida . write ( l + '\n ' )
10  res=commands . getoutput ( "mv "+arquivo _ orig+ ' _ '+ s t r ( s i z e )+ ' . clu resultado ' )
```

4.2. Calculating the dispersion of the clustering coefficient

The flow of execution of this method is similar to the previous, but with some particularities. One difference lies in the fact that this takes as input the data generated by the clustering coefficient calculation method, not a file in Fast format. However, the main reason for making changes in the source code of the method is that it has as output the average of the dispersion values of each of the graphs. In this case, it was necessary to change it so that each task performed on the grid did not generate as output the average but only the sum of these values. The average will be calculated later.

Initially the file generator script to be submitted on the grid performs the same procedures prior methods but with some differences. First, a division of input files is not performed, since these are not united results of the processing of the clustering coefficient calculation method. In this script, as can be seen in follow code the input files are listed (line 1) and stored (line 2) in a list. After the list is traversed and the input files (.clu extension) are filtered and added to the grid submission file (line 3).

```
1  lista_diretorio =commands . getoutput ( " l s " )
2  lista_separada= lista_diretorio . split ( '\n ' )
3  …
4  script_grid . write ( '\t '+ ' init :\ t '+ ' put '+ arq _ s _ ext + ' . clu ' + arq _ s _ ext + ' . clu\n '
       )
5  …
```

After the processing of tasks on the grid, another script will run (code below) which carries the process of uniting the results. That the resulting values of each file are added, and the end is averaged (scattering coefficient). After the calculated average is written to an output file (line 18).

```
1  total _ col0 =0.0
2  total _ col1 =0 .0
3  total _ linhas =0
4  for i in range ( 0 , n_arq ) :
5          arq _ nlin=open ( arquivo _ orig+ ' _ '+ str ( i )+ ' _ '+ str ( size )+ ' . clu ' , ' r ' )
6          nlin=arq _ nlin . readlines ( )
7          total _ linhas = total _ linhas +len ( nlin )
8          arq _ nlin . close ( )
9          arquivo=open (arquivo_orig+ '_'+ str (i)+ '_'+ str (size)+ ' _ '+ str ( jan )+ ' . s ' , ' r ' )
10         linhas=arquivo . readlines ( )
11         for l in linhas :
12                 ln= l . split ( ' ' )
13                 total _ col0 = total _ col0 + float ( ln [ 0 ] )
14                 total _ col1 = total _ col1 + float ( ln [ 1 ] )
15         arquivo . close ( )
16         res=commands . getoutput ("rm"+arquivo_orig+ '_'+ str ( i )+ ' _ '+ str ( size ) + ' _ '+
               str ( jan )+ ' . s ' )
17 saida=open ( arquivo _ orig+ ' _ '+ str ( size )+ ' _ '+ str ( jan )+ ' . s ' , 'w+ ' )
18 saida . write ( str ( total _ col0 / total _ linhas )+ ' '+ str ( total _ col1 / total _ linhas ) )
```

4.3. Determination promoters using a neural network

This implementation is based on a sequential algorithm already available, which was developed by doctoral student in Biotechnology Program at UCS, Scheila of Avila and Silva, using the R language[15]. This algorithm, the sequential version, has four linked bonds, as seen in code as follow. The outer loop (line 1) refers to the number of neurons in the hidden layer. The subsequent loop (line 2) refers to the network number. The third loop (line 3) refers to the number of times used in NN. The inner loop (line 4) refers to the number of lists initialized (input file). But the command block corresponds to the commands that perform the simulations NN.

```
1  …
2  for Noculto=1 to 8 do
3          for i =1 to 10 do
4                  for j =1 to 30 do
5                          for k=1 to 10
6                                  Bloco de comandos
7  …
```

Since each loop is independent can run each iteration (command block) of these bonds separately. To do this simply perform the iterations in the script responsible for generating the jobs to the grid uploaded files. Thus, it has been the execution of each iteration in different grid computer. In this implementation three

tests were performed. In the first of the two more external loops were divided so as to be distributed on the grid.

In this case the two inner loops are executed in each grid machine. In the second test only the innermost loop was kept to be run on each grid machine. So divided the three most external ties, to run in a distributed way. The last test was divided four bonds to run in a distributed way, thus keeping only the command block NB responsible for performing each grid machine. For the distribution of tasks was implemented a script written in the Python language, and in this code we has an excerpt of this script.

```
1   …
2   for oculto in range ( 1 , 9 ) :
3           for i in range ( 1 , 11 ) :
4                   for j in range ( 1 , 31 ) :
5                           for k in range ( 1 , 11 ) :
6                                   iteracoes = j*5
7                                   script_grid . write ( ' task :\n ' )
8                                   script_grid . write ( '\t '+ ' init :\ t '+ ' put
                                            TreinoRNDGProm_Sigma70Aleat1
                                            TreinoRNDGProm_Sigma70Aleat1\n ' )
9                                   script _ grid . write ( '\t\t '+ ' put
                                            TesteRNDGProm_Sigma70Aleat1
                                            TesteRNDGProm_Sigma70Aleat1\n ' )
10                                  …
11                                  script _ grid . write ( '\t\t '+ ' put script _ grid . sh
                                            script _ grid . sh\n ' )
12                                  script _ grid . write ( '\t\t '+ ' put Script . R Script . R\n '
                                            )
13                                  script _ grid . write ( '\t '+ ' remote : . / script _ grid . sh'
                                            + str ( oculto ) + ' ' + str ( i ) + ' '+ str (
                                            iteracoes ) + ' ' + str (K) + '\n ' )
14                                  script _ grid . write ( '\t '+ ' final : get arquivo_ '+ str (
                                            oculto ) + ' _ '+ str ( i ) + ' _ '+ str ( iteracoes ) + ' _
                                            ' + str ( k ) + ' . tar . gz ' + ' arquivo_ '+ str (
                                            oculto ) + ' _ '+ str ( i ) + ' _ '+ str ( iteracoes ) + ' _
                                            ' + str ( k ) + ' . tar . gz\n ' )
15  …
```

The algorithm execution R in each grid computer is done via a script written using the Shell Script language. This is based on measuring the number of bonds to be partitioned and the end that compresses the files obtained from the execution of the NN. These files have the NN hit rate, that is, values between 0 and 1, which correspond to the likelihood that the DNA sequence is a promoter or not. It is also generated the average quadratic error in order to verify the quality of the results. In follow code we have the code running on each grid machine.

```
1   /opt/R 2.9.0/ bin/R−−file = Script . R−−args $1 $2 $3 $4
2   tar −czf arquivo_$1_$2_$3_$4 . tar . gz *. t x t
```

5. Results

For testing the implementations was used the Griducs which consists of a existing institutional grid in the UCS. This is formed by approximately 300 computers not dedicated, there are, computers that have the primary purpose use in teaching activities in undergraduate and Graduate courses at UCS. These computers have 3 operating systems with the necessary tools to provide features for the OurGrid. To fulfill the entropy calculation methods, frequency, coefficient clustering and dispersion of the clustering coefficient were used as input all matching files Fast species of Kingdom Bacteria and Fungi Kingdom available at NCBI until the date of October 26, 2009[9]. For the prediction method of an NN by promoters

were used in all available promoters RegulonDB [3] associated with Sigma70 (sedimentation weight) to the same date.

For the analysis of developed implementations calculated the speed-up and efficiency of the developed implementations. The speed-up(S_p) aims to demonstrate how the parallel execution is faster than the sequential execution and its calculation is done by dividing the sequential time (T s) by the parallel time(T_p) as follows T_s/T_p. The ideal speed-up is achieved when your result is equal to the number of machines ($S_{p=p}$). Since the efficiency refers to the percentage of use of the processors in the execution, this value ranges from 0 to 1, and is calculated by dividing the speed-up (S_p) by the number of processors (P) used as follows

$S_{p/p}$.

In Figure 7 there is the sequential and the parallel long time methods. The sequential time is an estimate based on the sum of all tasks performed on the grid. We opted for the calculation of this estimate as the sequential execution of all methods with a single computer was impossible. In tests of entropy calculations, periodicity, and clustering coefficient were used 177 machines. As for calculating the dispersion of the clustering coefficient were used 142 machines. The number of machines used in these tests correspond to the number of machines available on the grid at the time of testing. For the prediction of promoters through an NN were used 80 machines since this was the maximum number of tasks.

For calculation of the entropy can be seen that the sequence time is a little higher than the parallel time. As can be seen in Figure 8, the speed-up obtained in this method is only 4.49, and as can be seen in Figure 9 has an efficiency of only 0.025. This is due to the fact that the time spent in the execution phase of each task, on average, of 1.3 seconds, since the data transfer time for each task is 2 seconds on average, so it has been a high transfer time compared to the runtime, which results in poor performance of the developed implementation.

In the method for calculating periodicity can be observed in Figure 7 that the time sequence is much higher than the parallel time, so there is a significant improvement in processing time. In Figure 8 it can be seen that the speed-up obtained in this method is 137.96, which means a great improvement during the parallel execution in the sequential relation. It can be seen from Figure 9 that have an efficiency of 0.779 which means that a high percentage of resources of each execution of the computer grid was used. This is due to the fact that this method uses 0.8 seconds of data transfer for each task, which is low compared to 205/2 used to perform each task.

As for calculating the clustering coefficient, as can be seen in Figure 7, there is a significant improvement in the time of the parallel processing method related sequence. In Figure 8 it can be seen that the speed-up of this method is 126.87. In the Figure 9 it can be seen that the efficiency is also high with a value of 0.716. In this case, each task processing time is 92 seconds, the data transfer time 1.2 seconds. For clustering coefficient dispersion method as viewed in Figure 7 the parallel execution time is greater than the sequential execution time. As can be seen in Figure 8.

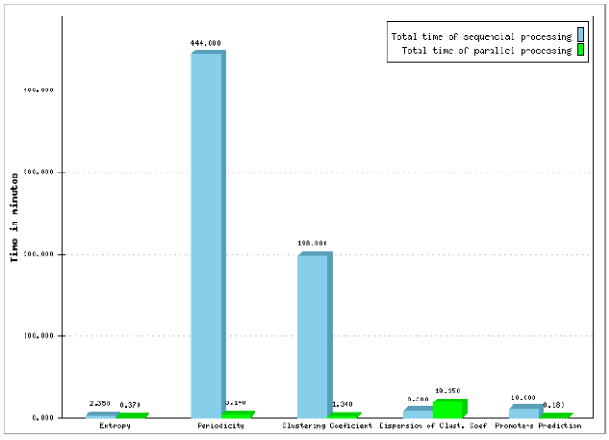

Figure 7: Comparation of sequencial and parallel processing

The speed up obtained in this method is only 0.43, and as can be seen in Figure 9 has an efficiency of

only 0.003. This can be explained because the data transfer time to each machine is 1.5 seconds, very high compared to only 0.6 second execution time for each task, so has a very high transfer time data. This implementation could be improved by running larger files, i.e. the division of the original Fast files in larger files. Thus, reducing the number of tasks execution and increasing processing time required for each task in the grid nodes.

For the promoters prediction method via a RN, as can be seen in Figure 7 the parallel execution time is considerably shorter than the sequential execution time. In Figure 8 and Figure 9 it can be seen that the speed-up and the efficiency of this method are 33.66 and 0.420, respectively. This method achieved good results because the data transfer time is 29 seconds, lower than execution, which is 45 seconds. In this method were made three tests. But two tests, the parallelization of loops 3 and the parallelization of loops 4 showed very high time to results, and in this case we chose to abort the execution. In this case, there is a file transfer time (sending and receiving) higher than the run time.

The prediction method of promoters through a NN was the subject of a search coordinated by Prof. Dr. Sergio Echeverrigaray UCS Biology Center. This research resulted in the participation of the undergraduate student Mauricio Adami Mariani, author of this work, in the seventeenth event Meeting of Young Researchers of UCS 2009 as young researcher. The research was titled: "Using Computational Grids for parallelization of a machine learning algorithm"[2]. This also had the participation of Prof. PhD Gunther JL Gerhardt, Prof. PhD Scheila of Avila and Silva and Prof. PhD. Andre Luis Martinotto.

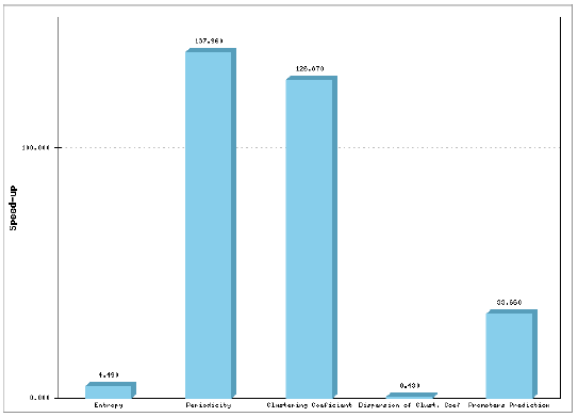

Figure 8: Speed-up of parallel codes

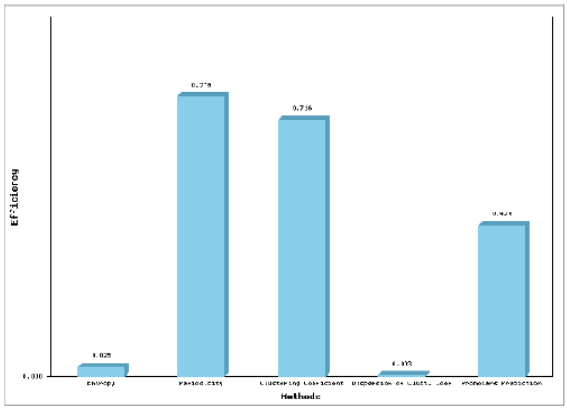

Figure 9: Efficiency of each parallel implementation

6. Conclusions

In this work we studied basics of molecular biology and bioinformatics methods. The area of biology has a large field of research and a lot of data to be processed. With the aid of these computing tasks can be accomplished in a more quick and efficient. This study was conducted with the aim of using these methods in a high performance environment. It was decided by computational grids such an alternative for

obtaining a high computing power at low cost. For parallelization of the calculation of entropy, calculation of frequency, calculation of clustering coefficient and calculation of dispersion of clustering coefficient was chosen split the Fast files used as input. As for the promoters prediction method through a NN implementation methodology consisted in the distribution of the sequential algorithm with ties.

Of the three parallel implementations of these developed implementations (frequency calculation, calculation of the clustering coefficient and prediction of promoters with an NN) were far faster than their respective sequential executions. An alternative to further reduce the processing time of the two methods in those(frequency calculation and calculation coefficient clustering) would be the increase in the computational grid customers. The third method that achieved improvement in their processing time (prediction of promoters with an NN), this improvement obtained by distributing two loops, creating 80 jobs, as mentioned in Section above, even a distribution of 3:04 ties that make up the sequential program, but these obtained very high processing times, if aborting them their executions.

Finally the methods calculation of entropy and dispersion of calculating of cluster coefficient did not achieve good results, with no significant improvements in their respective processing times or even lower performance. These longer used for transferring the data to the grid computers in the processing sequence.

6.1. Future works

This study aimed to assist research in bioinformatics UCS. From the developed implementations sought to reduce the processing time of some methods used in this field. Bioinformatics is an area that has several methods that require high-performance processing. So we can list some possibilities jobs future following:

- Adaptation of other methods of bioinformatics for HPCA.
- The GridUCS expansion through the use of computers that are not located in the UCS, but geographically dispersed or provided by volunteers.
- Creating a web environment that allows the availability of the tools developed so they can be used for real-time processing.
- Creating a web environment for the provision of the results generated by plays that work.

References

[1.] ANDRADE, N.;COSTA, L .;GERMÓGLIO, G.; CIRNE, W.; Peer-to-peer grid computing with the OurGrid Community. Universidade Federal de Campina Grande, Campina Grande, 2005.

[2.] ANDRADE, N.; CIRNE W.; BRASILEIRO, F.; OurGrid: An Approach to Easily Assemble Grids with Equitable Resource Sharing. Federal University of Campina Grande, Campina Grande, 2003.

[3.] BUYYA, R. High Performance Cluster Computing: Architectures and Systems. [S.l.]: Prentice Hall, 1999.

[4.] CIRNE, W.; NETO, E. S.; Grids Computacionais: Da Computação de Alto Desempenho a Serviços Sob Demanda. Fortaleza, CE: SBC, Federal University of Ceara, State University of Ceara, 2005.

[5.] DANTAS, M. Computacão distribuída de alto desempenho. Rio de Janeiro: Axcel Books, 2005.

[6.] FARACH, M.; NOORDEWIER M.; SAVARI S.; SHEPP L.; WYNER A.; ZIV J.; On the entropy of DNA: Algorithms and measurements based on memory and rapid convergence. Proc. 5th Annual ACM-SIAM symposium on Discrete Algorithms; ACM-SIAM, 1994.

[7.] FOSTER Ian. Designing and Building Parallel Programs. 1. ed. Reading, USA: Addison Wesley, 1995.

[8.] GERHARDT ,G. J.L.; LEMKE, N.; CORSO,G.;Network clustering coefficient approach to DNA sequence analysis; Paper published in Chaos, Solitons and Fractals, 2005;

[9.] GERHARDT ,G. J. L.; PANIS, R. J.; SANTOS, L. Presença de Períodos em Sequências de DNA. Caxias do Sul: UCS, Artigo não publicado, ano 2004.

[10.] GERHARDT, G. J. L., SILVA, S. de A.; PANIS, R. J.; SANTOS, L. dos; Correlações entre Cromossomos Usando Periodicidades em Seqüências de DNA. II Workshop of Technology in Computer Applied at Natural Enviroment; Publicado em 2004.

[11.] GIBAS, C.; JAMBECK, P.; Developing Bioinformatics Computer Skills; Estados Unidos, O'Reilly, 2001.

[12.] HAYKIN, S.. Neural networks: a comprehensive foundation. 2. ed. New Jersey: Prentice-Hall, 2005.

[13.] HORTA, J. O. C.; Estimadores de Entropia para Sequências de DNA; 2001; 144p. Dissertation (Masters dissertation in Computer Applied) - University of Sao Paulo, São Paulo.

[14.] JACOB, B.; BROWN, M.; FUKUI, K.; TRIVEDI, N. Introduction to Grid Computing. , 2005. Disponvel em: http://www.redbooks.ibm.com/abstracts/sg246778.html. Access in: october. 2009.

[15.] KONIGES, A. E.; Industrial Strength Parallel Computing; San Francisco, California, Morgan Kaufmann Publishers, 2000.

[16.] LESK, Arthur M., Introdução à Bioinformática, 2a Ed.- Porto Alegre: Artmed, 2008.

[17.] LEWIN, B.; Genes VII; 1.ed.; Porto Alegre, Artmed, 2001.

[18.] MARIANI, M. A. ,GERHARDT, G. J. L., SILVA, S. de A.;MARTINOTTO, A. L., ECHEVER-RIGARAY, S.; Utilização de Grids Computacionais para a Paralelização de um Algoritmo de Aprendizado de Máquina; XVII Encontro de Jovens Pesquisadores da UCS; 009.

[19.] MOUNT, David W. Bioinformatics : Sequence and Genome Analysis. New York: Cold Spring Harbor Laboratory, 2000.

[20.] NELSON, D. L.; COX, M. M.; Lehninger Princípios de Bioquímica.; 3.ed.; São Paulo, Sarvier, 2002.

[21.] Regulon DB. Disponvel em: <http://regulondb.ccg.unam.mx>. Access in: october. 2009.

[22.] RUSSELL, S.; NORVIG, P.; Inteligencia Artificial; Rio de Janeiro. Elsevier. 2004.

[23.] SANTOS, L. dos;Caracterização Computacional de Padrões Estruturais em Sequências de DNA Relacionadas a Processos em Redes Metabólicas; 2009; 127p.; Dissertation(Masters dissertation in Computer Applied) – National Institute of Space Research(INPE), São José dos Campos.

[24.] SEEFELD, K.; R For Bioinformatics; OREILLY & ASSOC; 2005

[25.] SHANNON, C.; WEAVER, W. The Mathematical theory of communication. 5 ed. Urbana/Chicago, Illinois State University Press, 1963

[26.] SILVA, S. de A.; Redes Neurais Artificiais Aplicadas na Caracterização e Predição de Regiões Promotoras; 2006; 144 p.; Dissertation (Masters dissertation in Computer Applied) - University of Vale dos Sinos(UNISINOS), São Leopoldo.

[27.] TANENBAUM, A. S.;Organização Estruturada de Computadores; Rio de Janeiro, RJ, Livros Técnicos e Científicos Editora, 2007.

[28.] VOET, D.; VOET J. G.; Bioquímica; 3.ed; Porto Alegre, Artmed, 2006.

[29.] WHEELER, David L.; BARRET, Tanya; BENSON, Dennis A.; BRYANT, Stephen H.; Canese, Kathi; CHETVERNIN, Vyacheslav; CHURCH, Deanna M.; DICUCCIO, Michael; EDGAR, Ron; FEDERHEN, Scott; GEER, Lewis Y.; HELMBERG, Wolfgang; KAPUSTIN, Yuri; KENTON, David L.; KHOVAYKO, Oleg; LIPMAN, David J.; MADDEN, Thomas L.; MAGLOTT, James Ostell; PRUITT, Kim D.; SCHULER, Gregory D.; SCHRIML, Lynn M.; SEQUEIRA, Edwin; SHERRY, Stephen T.; SIROTKIN, Karl; SOUVOROV, Alexandre; STARCHENKO, Grigory; SUZEK, Tubga O.; TATUSOV, Roman; TATUSOVA, Tatiana A.; WAGNER, Lukas; YASCHENKO, Eugene. Database resources of the National Center for Biotechnology Information. Nucleic Acids Research, v. 34, 2006.