2014 IEEE 14th International Conference on Bioinformatics and Bioengineering

Optimization and Analysis of Seriation Algorithm for Ordering Protein Networks

Felipe A. Kuentzer,
Alexandre S. Pereira,
and Alexandre M. Amory
Faculdade de Informática, PUCRS University,
Av. Ipiranga, 6681, Porto Alegre, Brazil

Gabriel Perrone,
Samoel R. M. da Silva,
João M. Dinis,
and Rita M.C. de Almeida
Instituto de Física, UFRGS University,
Av. Bento Gonçalves, 9500, Porto Alegre, Brazil

***Abstract*—Analysis by Transcriptogram was developed as a solution to reduce the noise in the microarray measuring technique of the Transcriptome, and has demonstrated potential to be applied as a method of disease diagnostics. The noise reduction in the measurement is achieved by ordering the proteins of a given protein interaction network in a linear way, allowing gene expression analysis in whole genome scale. The ordering process uses a seriation technique, which models a protein-protein association network into an undirected graph. So far, the viability of the diagnosis method was hindered by the high runtime of the ordering algorithm, since the *Homo sapiens* network can have around 30 thousand proteins. This paper presents some optimizations applied to the seriation problem, which lead to a significant reduction of execution time and the problem complexity analysed. Results show that the algorithm produces good results in reasonable time, e.g. order an undirected network of 9684 nodes in about 35 minutes, which is faster if compared with other seriation techniques evaluated.**

I. Introduction

After mapping and sequencing the of *Homo sapiens* genetic code (Human Genome Project [2]), one of the challenges of biological area becomes the analysis of these data, determining which genes are expressed in a given cell or particular pathological condition. Proteins are the main coordinators of biological functions in cells, but only determine which proteins are expressed is not enough to understand its biological role. A biological function requires several proteins acting coordinately, furthermore, a single protein can be part of many different cellular functions, thus forming an extremely complex network.

To determine the role of a given protein in the total context of a cell, in addition to the dozens of laboratory experiments to be performed, a laborious and complicated analysis of data generated is required. Due to the huge amount of information about proteins and their interactions, the classification of sets of proteins becomes a nontrivial task. In this context the Transcriptogram [1] has been proposed and used as part of a method for diagnostic and prevention of diseases at the cellular level [9].

The Transcriptogram technique involves a new method for ranking protein networks associated with Transcriptome analysis. Transcriptome is a set of RNA transcription present in the cell that contains all the information of which proteins will be produced in the cell. Among the techniques used to quantify the Trasncriptome there is the microarray technique [3]. One problem with this technique is the noise contained in the measure and the variability of the measurement, which presents differences depending on the lab that made the measures. Still is a very common technique, being a major source of such data.

Due to data noise and variability of the microarray technique, a new method for ordering protein networks emerged as part of the Transcriptogram. Ordering the protein network creates a ranked list that relates the proximity of two proteins with the probability of their association. Given this relationship it is possible to perform averages over the ordered regions applying the window modularity technique, thus smoothing the noise.

The ordering algorithm presented in this paper is called CFM (Cost Function Method) and can be classified as a seriation technique, which is an exploratory combinatorial data analysis method that reorders objects into a sequence to show regularity and patterns among the series of objects. Although the seriation goal is to maximize similarity between neighbouring objects, the exact objective function formulation and permutation method is still restricted to specific problem goal and dataset, and the fact that there is virtually hundreds of ways to define sameness and similarity. Moreover, some seriation techniques are limited by the dataset size, resulting in a high runtime or even impractical times.

As previously mentioned, protein networks have a large amount of information. For example, the *Homo sapiens* network used in this paper's experiments has about 9700 proteins and 160 thousand interactions. Not only the ordering method must support large networks, but it also must present a satisfying result in reasonable time to make the diagnostic method viable.

In this context, this paper presents an efficient implementation and the complexity analysis of the CFM algorithm, and a comparison to other known seriation methods. The rest of the paper is organized as follows: First a review about the seriation techniques is presented, followed by an overview about the concepts of window modularity and biological characterization, contextualizing the CFM's importance for the Transcriptogram. Then the CFM algorithm is described, it is presented the description of some simple data structures used to achieve an efficient implementation, and the algorithm's complexity is analysed. At the end, the CFM is compared with other seriation methods addressed in the literature, leading to

978-1-4799-7502-0/14 $31.00 © 2014 IEEE
DOI 10.1109/BIBE.2014.43

conclusions.

II. Seriation Background

Seriation is a technique of exploratory data analysis to rearrange objects in a linear order so that it reveals regularity and patterns among the whole series. As pointed in [11], the work of Petrie [13] represents the first systematic method for seriation and the work of Czekanowski [14] the first on matrix permutation visualization.

A seriation and matrix visualization result contains information about the clustering of data with additional information about how one cluster is related to another, which are the bridging objects and how are the transition of objects inside the cluster. In fact, seriation is closely related to clustering, although there is no clear agreement in the literature about their distinction [11]. According to [12], the main difference between seriation and clustering is that with seriation no information of any kind is lost, and the number of clusters does not have to be presumed. Nevertheless, similar problems have been addressed in the literature by using graph clustering methods [24].

One of the challenges concerning the seriation is to determine the objective function that best suits the problem. Another challenge is to define and evaluate which seriation method is the best. Since seriation is applied in several areas (e.g. Archaeology [13], Biology [16], Sociology [15] and Psycology [17]), and each one aims different goals, it is hardly possible to have a generic seriation method that suits all cases.

A seriation typically starts with a $N \times N$ symmetric dissimilarity matrix M. To reorder the objects in M a permutation function that optimizes the value of a given *loss* function or *merit* function is applied. A symmetric dissimilarity matrix is known as two-way one-mode data since it has columns and rows (two-way) but only represents one set of objects (one-mode) [19]. Although seriation can also be presented as two-way two-mode data, they are not relevant for this paper, since protein networks dealt with are two-way one-mode.

The state of the art in terms of seriation software includes *PermutMatrix* [18], *PAST* (PAlaeontological STatistcs) [20] and *Pajek* [21]. An alternative to these softwares is a specialized library for R language, the *R Package seriation* [19]. This library was used by [22] in order to compare their proposed dendrogram seriation algorithm and seriation criteria with known seriation techniques.

The *R Package seriation* provides an environment that allows to apply different seriation methods and evaluate input data with commonly employed objective functions. The package is also flexible to new objective functions and seriation methods, that can be easily registered to the framework. The seriation methods of [19] are a good representation of the seriation techniques typically used. The list of seriation methods are presented in Table I.

Each one of the methods tries to minimize or maximize, depending if it is a loss or merit function, one objective function. Details about these functions and references to the implemented methods can be found in [19]. As shown in Table I the objective function of some methods are not available.

The Table I also presents the input data. Implemented methods support either a *matrix* for two-way two-mode data or a *dist* (dissimilarity matrix) for two-way one-mode data. Different dissimilarity measures are also available and can be applied to the raw input data (e.g. Euclidean distance, Binary distance and Manhattan distance).

Regarding the objective functions, the *R Package seriation* features the loss or merit function presented in Table II. For the purpose of this paper, the *R Package seriation* will be used to compare the CFM performance against some of the available methods in the literature.

III. Seriation and Gene Expressions Analysis

As seen in Section II, seriation is widely used in different areas. In biological area seriaion was used, for instance, to improve the visualization and to find highly expressed genes in Cancer data [23] and Fibroblast data [24]. Following sections briefly presents the use of seriation in the context of the Transcriptogram method. Reference [1] is indicated for those who want more details about Transcriptogram.

A. Window Modularity

The final ordering produced by a seriation algorithm should clearly show distinct modules. For example, Figure 2 (detailed further later) shows different examples of ordering for the same network but different seriation algorithms. Modules are present in the form of black dots agglomerations around the adjacency matrix diagonal. This visual analysis is good but not enough to identify interactive modules. The modules identification uses the measure called *window modularity*.

This measure represents the ratio between the number of interactions of the nodes in a selected region of the ordering and the total number of interactions of these nodes. The resulting value ranges between 1 and 0, where 0 means that there is no interaction among the nodes in the selected region and 1 means that all the links of the vertices occur within the selected region.

B. Biological characterization

After using the window modularity to divide the network into modules, it is possible to identify the main biological processes associated with each module. Using the tool DAVID [6], the list of proteins of each module is provided and a list of biological processes is retrieved for each module from the Gene Ontology [7] database.

Once the main biological processes and their related proteins are known, it is possible to visualize the distribution of proteins that participate in each process within the entire network. This is essential to check whether the biological process corresponds mainly to a single module or if it is distributed among several.

Figure 1 explicitly shows the main motivation to have an efficient seriatiom algorithm. On one hand, with random protein positioning the chart becomes meaningless, and it is not clear where the modules are (gray peaks in the modularity). On the other hand, a seriated graph clearly shows how the same biological processes are more expressed in few modules.

TABLE I. CURRENTLY IMPLEMENTED SERIATION METHODS [19]

Algorithm	Acronym	Objective function	Input data
Simulated annealing	ARSA	Gradient measure	dist
Branch-and-bound	BBURCG	Gradient measure	dist
Branch-and-bound	BBWRCG	Gradient measure (weighted)	dist
TSP solver	TSP	Hamiltonian path length	dist
Optimal leaf ordering	OLO	Hamiltonian path length (restricted)	dist
Bond Energy Algorithm	BEA	Measure of effectiveness	matrix
TSP to optimize ME	BEA_TSP	Measure of effectiveness	matrix
Hierarchical clustering	HC	-	dist
Gruvaeus and Wainer	GW	-	dist
Rank-two ellipse seriation	Chen	-	dist
MDS first dimension	MDS	-	dist
First principal component	PCA	-	matrix

TABLE II. OBJECTIVE FUNCTIONS AVAILABLE IN THE R PACKAGE SERIATION [19]

Name	method	Function type	Input data
Anti-Robinson events	"AR_events"	loss	dist
Anti-Robinson deviations	"AR_deviations"	loss	dist
Gradient measure	"Gradient_raw"	merit	dist
Gradient measure (weighted)	"Gradient_weighted"	merit	dist
Hamiltonian path length	"Path_length"	loss	dist
Inertia criterion	"Inertia"	merit	dist
Least squares criterion	"Least_squares"	loss	dist
Measure of effectiveness	"ME"	merit	matrix
Stress (Moore neighbourhood)	"Moore_stress"	loss	matrix
Stress (Neumann neighbourhood)	"Neumann_stress"	loss	matrix

IV. COST FUNCTION METHOD

The CFM algorithm models the protein-protein interaction network of a given organism as an undirected graph of N nodes in an adjacency matrix M of size NxN. The matrix elements $M_{i,j}$ are 1 or 0 depending on whether or not the ith and jth proteins interact. The result is a symmetric matrix of zeroes and ones with a null diagonal.

Initially the nodes are distributed randomly in the adjacency matrix, as shown in Figure 2(a). In order to emphasize the interaction between the proteins, the CFM algorithm attempts to approximate the nodes that interact with each other more. This corresponds to approximate the cores to the matrix diagonal. The protein approximation is achieved by minimizing the cost function H presented in Equation 1, where $|..|$ is the positive difference of the matrix elements located at neighbouring sites and d_{ij} is the distance from the point (i, j) to the diagonal, that is, $d_{ij} = |i - j|$. The total cost H of the matrix M is the sum of the individual cost of nodes that represents a protein interaction and its neighbours.

$$H = \sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}(|M_{i,j} - M_{i+1,j}| + |M_{i,j} - M_{i-1,j}| + |M_{i,j} - M_{i,j+1}| + |M_{i,j} - M_{i,j-1}|) \quad (1)$$

After starting with the nodes randomly ordered, the algorithm proceeds by randomly choosing a pair of proteins within the range $[1, N]$ and swapping their position on the ordering and consequently changing the interaction matrix too. In the interaction matrix, this process corresponds to swap the columns and swap the rows that represents the chosen protein. Although the matrix have been modified, since the the entire columns and rows where moved, no information about the interactions is lost.

Once the columns and rows are swapped, the new positioning of the nodes alters the matrix configuration and its cost must be recalculated. The new cost H' is then compared with the previous cost H. If the difference of the costs $\Delta H = H' - H$ is negative, the new configuration is accepted, otherwise, the configuration can still be accepted with probability $P = exp(-\frac{\Delta H}{T})$, where T is a virtual temperature used by the Simulated Annealing algorithm [10], which is applied to avoid local minimum.

The initial temperature is kept constant while the swap process is repeated. Over time, T is reduced in a constant interval τ by the cooling factor γ to the point of being almost null. The τ interval corresponds to a number of Monte Carlo Steps (MCS). Each MCS corresponds to the number of random choices equals to the number of nodes in the system. For example, if the graph has 9000 nodes, it means that one MCS consists of 9000 attempts of swapping protein pairs. The execution ends when a number of MCS m is achieved.

V. CFM'S CPU TIME OPTIMIZATIONS

The following sections describes the optimizations for the two main parts of the algorithm: the swapping process and the cost calculation process.

A. Swapping pointers

One of the main bottlenecks of the algorithm is the swap part, which randomly selects two columns and rows to be relocated to another position in the adjacency matrix. The

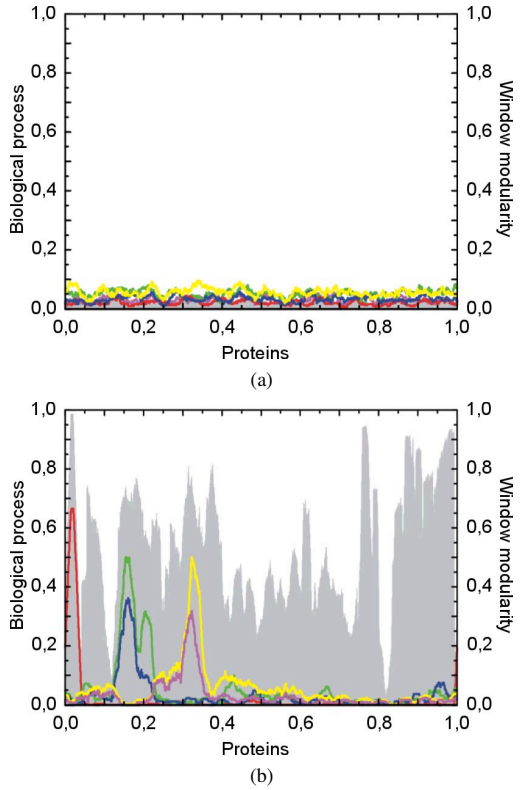


Fig. 1. Biological process distribution in (a) random ordering and (b) CFM ordering. In the background (gray) it is the modularity. The color lines represent different biological process like, for instance, cell cycle (yellow), organic acid metabolic (green) and sensory perception of chemical stimulus (red). [8]

swap is called at least once at each iteration, and is basically dependent on the number of network nodes N. For this case a vector C of size N describes the current position of the matrix elements. The vector translates the indexes i and j before a matrix element is accessed, which leads to the element's current location $M_{C[i],C[j]}$. Thus, rather than manipulating the data matrix, swapping only updates the indexes stored in vector C.

B. Diagonal cost calculation

The protein networks used in this paper always produce a symmetric matrix. This characteristic allows to reduce by half the number of nodes that need to be traversed during the cost calculation process. Instead of traverse all position in the NxN matrix whenever a new matrix configuration is produced, the cost calculation can traverse half of the nodes, and at the end, the cost can be multiplied by two, so the result remains the same as the one without the optimization.

C. Partial cost calculation

The partial cost calculation recalculates the cost for the columns and rows swapped and their immediate left and right neighbours. In the worst case it means that six columns and six rows with *N* elements will be traversed.

This optimization requires that the matrix is fully traversed only at the beginning to calculate the initial cost. After the initial cost is determined the algorithm starts to call the partial cost function. The partial cost function is called before a swap and after a swap. Once a pair of nodes is randomly chosen, their contribution to the current total cost H must be calculated, which is the partial cost Hp. Then the swap process is executed and then a new partial cost Hp' is calculated. The new partial cost is the new contribution of the modified columns and rows. At the end, new total cost is $H' = H - Hp + Hp'$.

D. Sparse matrix representation

The last optimization exploits the fact that the protein networks produces an adjacency matrix that is sparse, and the number of ones elements is much smaller than the zeroes elements.

At first the idea was to eliminate completely the adjacency matrix and replace it by a new representation that only describes the interaction elements, or simply the ones of the adjacency matrix. This new representation was faster while traversing all the interaction, but demanded more time looking for interactions in its neighbours. The solution was to keep the adjacency matrix to observe the neighbours while traversing directly to the ones elements with the new representation. The new representation consists in a list of N vectors, each one describing the position of the ones elements in a given row.

VI. Complexity Analysis

The CFM algorithm is mainly divided in three functions: the *swap* function, the *getPartCost* function and the main loop. The CFM's main loop has two nested loops. The outer loop iterates through the number of steps m, while the internal one is dependent on the number of nodes N. The main loop controls the random choices of nodes, calls the *swap* function with the chosen nodes, calls the cost function so that the new configuration cost is calculated and verifies if the swap is accepted.

To perform a swap with selected nodes a and b, the *swap* function simply updates the pointer vector ($C[a] \leftrightarrow C[b]$).The complexity is constant and is defined by $O(1)$. In the partial cost function *getPartCost*, for each pair of chosen nodes a and b, the function searches by interactions of these nodes, its left and its right neighbours. Thus, the outer loop searches in three lists for column a, three lists for column b, three lists for row a and three lists for row b. In total, the outer loop searches interactions in only twelve lists, instead of the entire matrix (N^2) as the original algorithm.

The inner loop of the partial cost function iterates only over the existing protein interactions, i.e. where the matrix has non-zero values (nz). Thus, the complexity of the partial cost function shall be determined by nz, and not N. Recall that, since protein networks are typically sparse, the majority of the nodes have few connections, thus, $nz.N << N^2$. In addition to this, the partial cost function also traverses only half of the nz elements of the matrix. The diagonal optimization does not reduce the computation complexity since it divides the number of traversed items by a constant value (2). However, in practice, it helps to speed up the execution time. Finally, the complexity of the partial cost function can be stated as presented in Equation 2, where each term $\frac{3nz}{2}$ represents the

TABLE III. CPU TIME FOR DIFFERENT SERIATION METHODS.

Seriation Method	CPU Time (s)
ARSA	-
TSP	28833
OLO	14410
HC	14206
GW	14252
Chen	118074
MDS	15619
CFM	**2974**

time to transverse row a, row b, column a and column b. When simplified it can be expressed as $O(nz)$.

$$f(nz) = \frac{3nz}{2} + \frac{3nz}{2} + \frac{3nz}{2} + \frac{3nz}{2} = 6nz \qquad (2)$$

The main loop calls the *getPartCost* in two moments, so that the contribution of the randomly selected nodes to the overall cost is defined before and after the swap process. As mentioned earlier, the outer loop is dependent on m and the and inner loop is dependent on N. Since the *getPartCost* complexity is $O(nz)$, the CFM complexity can be expressed as $O(m.N.nz)$.

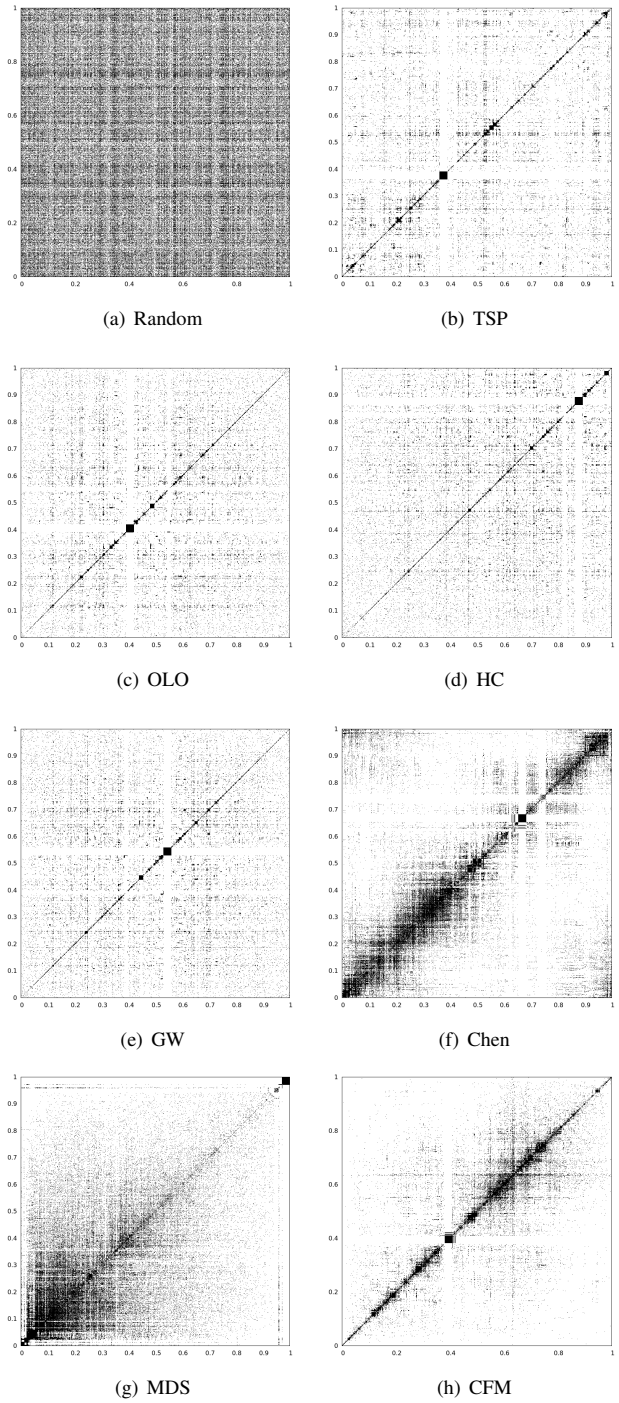
VII. PERFORMANCE ANALYSIS

The methods were selected from the *R Package seriation* based on the input data type and the size supported. As mentioned earlier, the protein networks used in this work are two-way one-mode data. So, only the methods that use a *dist* input data were selected. Among these methods, two of them were removed from the test scope, the BBURCG and the BBWRCG. Both methods are limited to problems around 30 nodes [19].

For this analysis the *Homo sapiens* protein-protein network was chosen and extracted from the STRING database v9.05 with a reliability score of 0.8 (details about the STRING database see Appendix). The network has 9684 proteins and 163509 interactions. To perform the tests a personal computer with an Intel Xeon W3670 3.20MHz, 12GB of RAM running a Linux Ubuntu 12.04 64 bits operational system was used. The same random initial order is applied to all methods. The dissimilarity matrix needed as input data to the seriation package is calculated using a Binary distance due to the characteristics of the raw input, which is a binary matrix.

The test with the ARSA method was stopped after four days of execution, therefore it is not present in Table III. The results with the seriation package methods show an execution time above 14000 seconds, while the CFM is under 3000 seconds.

Table IV presents results of several seriation methods with different optimization criteria. The results show that CFM outperforms all the methods in all optimization criteria, except for *Path_length* (PL) criterion. According to [22] the algorithms that minimizes the PL cost function are not concerned with the global structure of the dissimilarity matrix and they are only concerned with the structure just off the main diagonal. On the other hand, CFM is meant for global optimization.



(a) Random (b) TSP (c) OLO (d) HC (e) GW (f) Chen (g) MDS (h) CFM

Fig. 2. Initial random order and the final ordering generated for the different seriation methods.

Figure 2 illustrates the ordering generated by the different seriation methods. One can observe that all methods show a similar group, a dense square placed in the matrix diagonal, but visually CFM grouped more densely the protein interactions around the matrix diagonal.

TABLE IV. SERIATION PERFORMANCE FOR DIFFERENT METHODS USING DIFFERENT OPTIMIZATION CRITERIA. THE BEST RESULT FOR EACH CRITERIA IS HIGHLIGHTED IN BOLD.

	AR_events (loss)	**AR_deviations** (loss)	**Gradient_raw** (merit)	**Gradient_weighted** (merit)	**Path_length** (loss)	**Inertia** (merit)	**Least_squares** (loss)
Initial	12891086027	983328283.806	-10284444	-6557961.776	9654.367	1460848444256310	1465173562343840
CFM	**4981068587**	**141434359.763**	**12092239079**	**1261323512.553**	6288.950	**1465345662358760**	**1465171871835210**
TSP	9876188210	283303278.143	4608888908	1045143077.921	**4627.559**	1464328017393360	1465172160075810
OLO	9285319074	300499700.478	5435078841	1018690145.219	4676.345	1464479667265400	1465172195346310
HC	10129899457	323747884.564	4195093427	984046407.169	4862.422	1464329829195710	1465172241538020
MDS	6891024607	201279004.969	9078623362	1168255753.520	7469.688	1465142476667000	1465171995925830
GW	9420708526	311296313.253	5226244871	1002198985.086	4725.865	1464428923138910	1465172217334500
Chen	6158650972	165469250.357	10331791053	1225348340.410	7329.964	1464957040610610	1465171919802360

VIII. CONCLUSION

In summary, this paper presented an optimized seriation method, called CFM with complexity $O(m.N.nz)$. This paper also compared the CFM with some methods described in the literature. The CFM outperforms all analysed seriation method in terms of CPU time and also outperforms these methods using different criteria. Moreover, CFM provides the most compact nodes around the matrix diagonal.

Despite the good performance, we believe that it is still possible to improve the CFM performance by exploring parallelism with multi-core architectures and GPUs (Graphics Processing Unit).

The CFM algorithm here described can be downloaded at the homepage https://corfu.pucrs.br/redmine/projects/seriation. The installation package includes the source code, the binary file and the *Homo sapiens* network used in this paper's experiments. Also at the homepage there is a manual for the CFM algorithm, links to related works and other datasets.

APPENDIX
STRING DATABASE

Protein-protein interaction networks are retrieved from the STRING database [4] [5] (Search Tool for the Retrieval of Interacting Genes/-Proteins), and can be accessed at http://string-db.org. It provides information about the direct physical interactions between proteins or indirect (metabolic-routes), called functional, where two proteins that do not interact directly contribute to a given cellular process. Besides the direct and indirect interactions, other types of associations are provided by the STRING: neighbourhood, co-occurrence, fusion, co-expression, experiments and textmining.

Because of the large space of associations types provided by the STRING, it is likely that not all occur effectively. However, an interesting part of this database is the ability to control the quality and the information source. Equation 3 is applied in order to control the quality of each protein association. In the STRING, all interactions receive a reliability score for each association type (S_i), allowing to get a desired level of reliability (S), thus balancing the number of false positives and false negatives.

$$S = 1 - \prod_i (1 - S_i) \tag{3}$$

For this paper, the protein network were extracted from the STRING database v9.05, and have a combined score of 0.8, which means 80% of reliability. This score combines all mentioned association types except the textmining, which was not used to determine the reliability score of the protein associations.

REFERENCES

[1] Rybarczyk-Filho, J. L. L., Castro, M. A., Dalmolin, R. J., Moreira, J. C., Brunnet, L. G., de Almeida, R. M.: Towards a genome-wide transcriptogram: the saccharomyces cerevisiae case. Nucleic Acids Research 39-8, 3005-3016 (2011)

[2] of Energy Office of Energy Research, U. S. D., of Energy Office of Health, U. S. D., Research, E.: Human genome: program report (1990)

[3] Schena, M., Shalon, D., Davis, R. W., Brown, P. O.: Quantitative monitoring of gene expression patterns with a complementary DNA microarray. Science 270-5235, 467-470 (1995)

[4] Franceschini, A., Szklarczyk, D., Frankild, S., Kuhn, M., Simonovic, M., Roth, A., Lin, J., Minguez, P., Bork, P., von Mering, C., Jensen, L. J.: String v9.1: protein-protein interaction networks, with increased coverage and integration. Nucleic Acids Research 41, 808-815 (2013)

[5] von Mering, C., Jensen, L. J., Snel, B., Hooper, S. D., Krupp, M., Foglierini, M., Jouffre, N., Huynen, M. A., Bork, P.: String: known and predicted protein-protein associations, integrated and transferred across organisms. Nucleic Acids Research 33, 433-437 (2005)

[6] Dennis, G., Sherman, B. T., Hosack, D. A., Yang, J., Gao, W., Lane, H. C., Lempicki, R. A.: David: Database for annotation, visualization, and integrated discovery. Genome Biology 4(5):P3 (2003)

[7] Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., Harris, M. A., Hill, D. P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J. C., Richardson, J. E., Ringwald, M., Rubin, G. M., Sherlock, G.: Gene ontology: tool for the unification of biology. Nature Genetics, 25 ,25-29 (2000)

[8] Rybarczyk-Filho, J. L. L.: Medidas de performance metabólica usando a expressão gênica de genoma completo. PhD Dissertation. UFRGS, Porto Alegre, Brazil (2011) (in portuguese)

[9] da Silva, S. R.: A eficiência do transcriptograma. Masters Thesis. UFRGS, Porto Alegre, Brazil (2013) (in portuguese)

[10] Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P.: Optimization by simulated annealing. Science 220, 671-680 (1983)

[11] Liiv, I.: Seriation and matrix reordering methods: An historical overview. Statistical Analysis and Data Mining 3, 70-91 (2010)

[12] Spth, H.: Cluster analysis algorithms for data reduction and classification of objects. Ellis Horwood, Chichester, UK (1980)

[13] Petrie, W. M. F.: Sequences in Prehistoric Remains. The Journal of the Anthropological Institute of Great Britain and Ireland 29, 295-301 (1899)

[14] Czekanowski, J.: Zur Differentialdiagnose der Neandertalgruppe. Korrespondenzblatt Deutsch Ges Anthropol Ethnol Urgesch XL, 44-47 (1909)

[15] Forsyth, E. and Katz, L.: A matrix approach to the analysis of sociometric data: preliminary report. Sociometry 9, 340347 (1946)

[16] Hartigan, J. A.: Direct Clustering of a Data Matrix. Journal of the American Statistical Association 67, 123129 (1972)

[17] Hubert, L. J.: Problems of seriation using a subject by item response matrix. Psychol Bull 81, 976983 (1974)

[18] Caraux, G. and Pinloche, S.: PermutMatrix: a graphical environment to arrange gene expression profiles in optimal linear order. Bioinformatics 21, 12801281 (2005)

[19] Hahsler, M., Hornik, K. and Buchta, C.: Getting Things in Order: An Introduction to the R Package seriation. Journal of Statistical Software 25 134 (2008)

[20] Hammer, Ø.: PAST - PAlaeontological STatistics. (2013) http://folk.uio.no/ohammer/past/ (Retrieved May 1, 2013).

[21] de Nooy, W., Mrvar, M., Vladimir, B.: Exploratory Social Network Analysis with Pajek - Part of Structural Analysis in the Social Sciences. 2nd Edition, W. H. Freeman and Company, New York (2011)

[22] Earle, D.: Dendrogram seriation in data visualisation: algorithms and applications. PhD thesis, National University of Ireland Maynooth (2010)

[23] Khan, J., Wei, J. S., Ringnér, M., Saal, L. H., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Antonescu, C. R., Peterson, C. Meltzer, P. S.: Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. Nature Medicine 7(6), 673-679 (2001)

[24] Iyer, V. R., Eisen, M. B, Ross, D. T., Schuler, G., Moore, T., Lee, J. C. F, Trent, J. M., Staudt, L. M., Hudson, J., Boguski, M. S., Lashkari, D., Shalon, D., Botstein, D., Brown, P. O.: The transcriptional program in the response of human Fibroblasts to serum. Science, 283, 83-87 (1999)

[24] Schaeffer, S. E.: Graph clustering. Computer Science Review, 1, 27-64 (2007)