35th Annual International Conference of the IEEE EMBS
Osaka, Japan, 3 - 7 July, 2013

Efficient Markov Clustering Algorithm for Protein Sequence Grouping*

László Szilágyi[1] and Sándor M. Szilágyi[2]

Abstract— **In this paper we propose an efficient reformulation of a Markov clustering algorithm, suitable for fast and accurate grouping of protein sequences, based on pairwise similarity information. The proposed modification consists of optimal reordering of rows and columns in the similarity matrix after every iteration, transforming it into a matrix with several compact blocks along the diagonal, and zero similarities outside the blocks. These blocks are treated separately in later iterations, thus reducing the computational burden of the algorithm. The proposed algorithm was tested on protein sequence databases like SCOP95. In terms of efficiency, the proposed solution achieves a speed-up factor in the range 15-50 compared to the conventional Markov clustering, depending on input data size and parameter settings. This improvement in computation time is reached without losing anything from the partition accuracy. The convergence is usually reached in 40-50 iterations. Combining the proposed method with sparse matrix representation and parallel execution will certainly lead to a significantly more efficient solution in future.**

I. INTRODUCTION

Establishing protein families in large databases is one of the fundamental goals of functional genomics. A successful classification may contribute to the delineation of functional diversity of homologous proteins, and can provide valuable evolutionary insights as well [1]. By definition, protein families represent groups of molecules with relevant sequence similarity [2]. Members of such protein families may serve similar or identical biological purposes [3]. Identifying these families is generally performed by clustering algorithms, which are supported by pairwise similarity or dissimilarity measures. Well established properties of some proteins in the family may be reliably transferred to other members whose functions are not well known [4].

TRIBE-MCL is an efficient clustering method based on Markov chain theory [5], introduced by Enright et al [1] for protein sequence grouping. TRIBE-MCL assigns a graph structure to the protein database in such a way that each protein has a corresponding node, while initial edge weights in the graph represent computed pairwise similarity values, obtained via BLAST search methods [6]. Clusters were then obtained by alternately applying two operations to the similarity matrix: inflation and expansion.

In a previous paper [7] we proposed a modification of the TRIBE-MCL algorithm, in order to enhance its accuracy, improve its reliability and slightly weaken its computational load. These were achieved by introducing a time varying inflation rate and thus forcing the algorithm to apply a stronger inflation in the first iterations when the hard structure of the clusters is established, and reducing inflation strength for fine tuning the cluster shapes in later iterations. The acceleration of the computations was served there by a singleton filter, which excluded isolated nodes from the similarity graph that had no influence on other nodes in later iterations. In this paper we introduce a more efficient approach aimed at accelerating the execution speed of the algorithm, without damaging the outcome of the clusters. This is achieved with an optimal transformation of the similarity matrix, applied in the first iterations of the algorithm, when the hard structure of the partition is established.

The remainder of this paper is structured as follows: Section II takes into account the functional details of the TRIBE-MCL algorithm and presents our motivations to introduce modifications. Section III presents the details of the proposed efficient TRIBE-MCL algorithm. Section IV evaluates and discusses the efficiency of the proposed method. Section V presents the conclusions and gives some hints for further research.

II. BACKGROUND

TRIBE-MCL is an iterative algorithm, which operates on a directional graph. Each of the n nodes of the graph represents a protein sequence from the set we wish to cluster, while each edge length s_{ij}, $i, j = 1 \dots n$, shows the similarity between protein sequences of index i and j, respectively. Edge lengths are stored in the similarity matrix $S \in \mathcal{M}_{n \times n}$. Initial edge lengths usually come from pairwise sequence alignment. During the iterations, S behaves as a column stochastic matrix, whose elements represent probabilities of transitions (evolution).

The TRIBE-MCL algorithm consists of two main operations, namely the inflation and expansion, which are alternately repeated until a convergence is reached, that is, the similarity matrix becomes invariant during a cycle:

1) Inflation has the main goal to differentiate among connections within the graph, by favoring more likely direct walks along the graph in the detriment of less likely walks. It is computed via taking each element of the similarity matrix to the power of $r > 1$. The strength of this differentiation is controlled by

*Research supported by the Hungarian National Research Funds (OTKA), Project no. PD103921, and the Hungarian Academy of Sciences through the János Bolyai Fellowship program.

[1]L. Szilágyi is with Dept. of Electrical Engineering, Sapientia University, Calea Sighişoarei 1/C, 540485 Tîrgu Mureş, Romania (phone: +40-265-206-210; fax: +40-265-206-211; e-mail: lalo at ms.sapientia.ro) and with Dept. of Control Engineering and Information Technology, Budapest University of Technology and Economics, Magyar tudósok krt. 1, 1117 Budapest, Hungary (phone: +36-1-463-4027; fax: +36-1-463-2699).

[2]S. M. Szilágyi is with Dept. of Mathematics-Informatics, Petru Maior Univesity, Str. N. Iorga nr. 1, 540088 Tîrgu Mureş, Romania (phone/fax: +40-265-262-275; e-mail: szsandor72 at yahoo.com).

978-1-4577-0216-7/13/$26.00 ©2013 IEEE

the so called inflation rate r: large values express the preference of likely walks more severely, causing sudden ruptures within the graph, possibly not in the ideal place. Low inflation rates are more likely to yield smooth partitions, but the convergence may become rather slow.

2) Expansion operation is intended to reveal possible longer walks along the graph, to emphasize changes within the protein structures that happened in two or more evolutionary steps. Expansion is achieved via matrix multiplication, by taking similarity matrix S to the second power.

Auxiliary computations are also included in each iteration, in order to maintain the similarity matrix S as a symmetric column stochastic matrix.

Clusters are defined as connected subgraphs within the graph described by the similarity matrix, so a stable state of the similarity matrix means that the clusters don't change their contents during an iteration.

In a previous paper [7], we have proposed a series of generalizations of the conventional version of the TRIBE-MCL algorithm [1], e.g. time-variant inflation rate, generalized inflation scheme, singleton filter, etc. These changes brought slight improvements to the accuracy and efficiency of the algorithm.

III. MATERIALS AND METHODS

In this paper we introduce an efficient formulation of the TRIBE-MCL algorithm, with the aim of seriously reducing its computational load, without harming the accuracy of the partitioning. We will test the proposed method on the proteins of the SCOP95 database.

A. The SCOP95 Database

The SCOP (Structural Classification of Proteins) database contains protein sequences in order of tens of thousands, hierarchically classified into classes, folds, superfamilies and families [8]. The SCOP95 database involved in this study, is a subset of SCOP (version 1.69), which contains 11944 proteins, exhibiting a maximum similarity of 95% among each other. Pairwise similarity and distance matrices (BLAST [6], Smith-Waterman [9], Needleman-Wunsch [10], PRIDE [11], etc.) are available at the Protein Classification Benchmark Collection [12]. In this study we employ BLAST similarity measures, because that one suppresses low similarities, thus contributing to computational load reduction.

B. Cluster Separation

Most of the computational load of the algorithm is caused by the matrix multiplication, which has a theoretical complexity of $\mathcal{O}(n^3)$. In order to reduce runtime, it would be beneficial at any time of the execution, to separate those proteins which no longer have any influence upon the others. This idea we employed in the previous paper [7], where we proposed to exclude the rows and columns of singletons from the similarity matrix in each iteration. This way we achieved $30\%-50\%$ reduction of the overall processing time, depending on the percentage of singletons within the data.

In the following, we will formulate a more optimal separation scheme of clusters. Let us denote by Σ the initial set of proteins, which is intended to be classified. At any iteration t, we may look for connected subgraphs in the graph represented by the similarity matrix S. Whenever we find a subset of proteins $\Sigma_1 \subset \Sigma$, corresponding to a connected subgraph, isolated from the rest of the proteins ($s_{ij} = 0$, $\forall i \in \Sigma_1$ and $j \in \Sigma \setminus \Sigma_1$), in further iterations we may treat the proteins of Σ_1 separately from the others, because the rows and columns of S corresponding to these proteins will not interact with any other rows and columns. If we reorder all rows and columns of the similarity matrix S such that connected and isolated subgraphs are placed in consecutive rows and columns, we will have a similarity matrix formed by small square shaped blocks of nonzero elements placed along the main diagonal, and all other elements of the matrix will be zero.

In order to implement this idea, we need to define a reordering buffer R of size n, which will contain the grouped protein indexes corresponding to connected and isolated subgraphs in the graph represented by S. Further on, we need a group buffer Q to store the indexes of initial elements of protein groups within the reordering buffer. The latter buffer will need a time-variant size of storage (denoted by q), but it will never exceed the limit of n items.

Identifying the connected subgraphs is performed as follows:

- Initialize the count of found nodes $m = 0$, found subgraphs $q = 0$, map (set) of found nodes $M = \Phi$.
- While there exists node in the graph not yet mapped ($m < n$), find next such node having index i. Increment q, and let $Q_q = i$; increment m, add element m to set M and let $R_m = i$. Recursively search graph for nodes j connected to node i (with criterion $s_{ij} > 0$), include all found nodes into the set M, incrementing counter m accordingly, and inserting each j into buffer R ($R_m = j$).
- When the above loop ends, m should be equal to n, and M contains all nodes, because they are all found. The output of this step is the ordered buffer R, and the group buffer Q indicating the q separated groups.

Having the isolated groups of nodes separated, we may reformulate the operations performed within each iteration as follows. For each square block along the diagonal of reordered matrix S, that is, for each $b \in \{1, 2, \ldots, q\}$, we consider the subset of proteins in the connected subgraph $\Sigma_b = \{R_{Q_b}, R_{Q_b+1}, \ldots, R_{Q_{b+1}-1}\}$ assuming that $Q_{q+1} = n + 1$, and then

- inflation is computed as:

$$s_{\alpha\beta}^{(\mathrm{new})} = \left(s_{\alpha\beta}^{(\mathrm{old})}\right)^r \qquad \forall \alpha, \beta \in \Sigma_b, \tag{1}$$

- expansion is given by the formula:

$$s_{\alpha\beta}^{(\mathrm{new})} = \sum_{\gamma \in \Sigma_b} s_{\alpha\gamma}^{(\mathrm{old})} s_{\gamma\beta}^{(\mathrm{old})} \qquad \forall \alpha, \beta \in \Sigma_b, \tag{2}$$

- normalization is given by:

$$s_{\alpha\beta}^{(\text{new})} = s_{\alpha\beta}^{(\text{old})} \left(\sum_{\gamma \in \Sigma_b} s_{\gamma\beta}^{(\text{old})} \right)^{-1} \qquad \forall \alpha, \beta \in \Sigma_b, \tag{3}$$

- symmetry is approximated as:

$$s_{\alpha\beta}^{(\text{new})} = s_{\beta\alpha}^{(\text{new})} = \sqrt{s_{\alpha\beta}^{(\text{old})} s_{\beta\alpha}^{(\text{old})}} \tag{4}$$

  $\forall \alpha, \beta \in \Sigma_b$, $\alpha < \beta$. After symmetrization, similarity values below ε are reduced to 0.

The proposed algorithm is summarized as follows:

1) Define input protein set Σ.
2) Compute (or load) the initial similarity matrix S.
3) Set the parameters of the algorithm: inflation rate $r > 1$ and threshold $\varepsilon \in [10^{-4}, 10^{-3}]$.
4) Initially assume, that the whole graph is connected: initialize reordering buffer $R_i = i \ \forall i = 1 \ldots n$, and group buffer $Q_1 = 1$ and $q = 1$.
5) Inflate the similarity matrix according to the inflation rule described in Eq. (1), using inflation rate r.
6) Symmetrize and normalize the similarity matrix according to Eqs. (3) and (4), neglecting elements smaller than the threshold ε.
7) Identify connected and isolated subgraphs in the graph represented by S. Update buffers R and Q accordingly.
8) Expand the similarity matrix using Eq. (2).
9) Repeat steps 5-8 until matrix S and its connected subgraphs stabilize.

C. Efficient and Parallel Execution

At the beginning, almost the whole set of input data is situated within a connected graph. In each iteration, the inflation suppresses low valued (but non-zero) similarity values, which makes connections disappear, thus causing ruptures within the graph. Whenever a subset of nodes gets isolated from others, it will never get connected again. This is assured by the nature of the performed operations. By reordering the rows and columns of the similarity graphs, we obtain blocks of non-zero values along the diagonal of matrix S. What is also important to remark here is that data situated in separate blocks will not have any influence on each other anymore during the execution. Theoretically, if we divide the n input data into ν equal blocks, then the execution time of the expansion reduces ν^2 times. The other operations preformed in each iteration also profit from this reformulation, and the extra work to be performed, namely the identification of isolated clusters, is very effective.

Separated blocks within the similarity matrix can be processed in separate parallel threads, although this is not yet implemented. The final clusters obtained by each thread should be merged after parallel processing. Parallel threads have the possibility to execute for each separate block only the necessary amount of iterations to reach the convergence, eliminating unnecessary iterations for quickly converging blocks.

IV. RESULTS AND DISCUSSION

The main goal of protein clustering is to reveal hidden similarities among proteins. When evaluating the accuracy of the output, one can count the number of mixed clusters (those which contain proteins from two or more different families) and their cardinality. We have shown in the previous work [7], that the inflation rate is the main factor to influence the amount of mixed clusters. The approach proposed here computes exactly the same partitions as the conventional TRIBE-MCL, in a more efficient way. That is why the evaluation of accuracy is unnecessary in this study. The reader interested in accuracy details is referred to [7]. All efficiency tests were run on PC with quad core Intel i5 processor running at 2.53GHz frequency, and 4GB RAM.

We have employed the proposed algorithm to classify either the whole set of 11944 proteins in the SCOP95 database, or selected subsets. At the selection of subsets, whole families were chosen from the hierarchical data structure, in order to keep all connections of each selected protein. The hierarchical structure of the SCOP database was only used to select input data and verify the final partition accuracy. Partitioning only uses the pairwise similarity data.

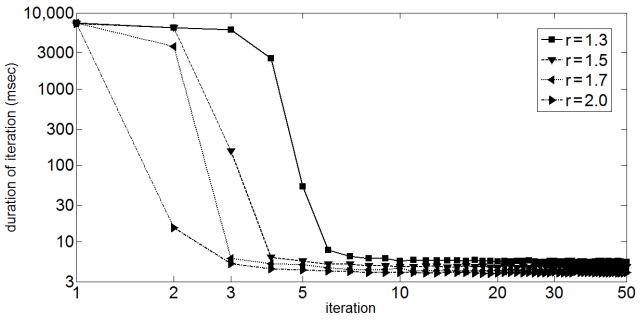

Fig. 1. The duration of the first 50 iterations, using the proposed method at various inflation rates, to classify 908 proteins from SCOP95

Fig. 1 summarizes some efficiency tests performed on a set of 908 proteins (all families from SCOP95 which have 11 to 14 proteins): varying the inflation rates between 1.3 and 2.0, the duration of each iteration was recorded and plotted in this figure. In only 4-6 iterations, the large connected block within the similarity graph is broken into small subgraphs, enabling us to compute subsequent iterations on small matrices. Late iterations are performed approximately 1000 times quicker. Although the computation load stabilizes at a low level after the initial five iterations, the convergence of the output data requires around 40-50 cycles. Without this proposed efficient scheme, all iterations would need the same amount of computations as the first one. This way we are able to approximate the speed-up ratio reached via fragmenting the similarity matrix.

Figs. 2 and 3 present efficiency results of the proposed method on various data sets, using a fixed inflation rate $r = 1.5$. Fig. 2 indicates the duration of each iteration, while Fig. 3 plots the cardinality of the largest connected subgraph in the similarity graph. In every case, the two characteristics

correlate in the sense, that squaring larger matrices requires a lot more time. The other operations only have slight effects on execution time. Data sets involved in the tests reported here were chosen as all protein families with cardinality between 10-15 (1288 proteins), 10-20 (2106 proteins), 8-30 (3887 proteins), 5-50 (6522 proteins).

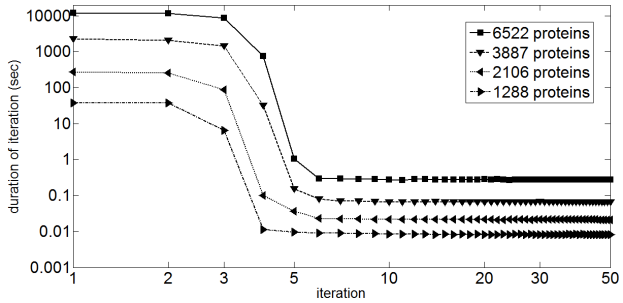

Fig. 2. The duration of the first 50 iterations, using the proposed method with various input data sets, plotted on logarithmic scale, using inflation rate $r = 1.5$

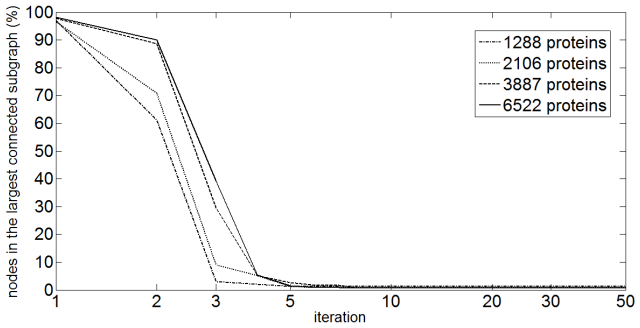

Fig. 3. The size of the largest connected subgraph after each of the first 50 iterations, for various input data sets, using inflation rate $r = 1.5$

Let us remark some trends identified from Figs. 1-3:

1) In every case, we needed a few iterations to break the similarity graph into several small isolated subgraphs. The larger the input data set, the more iterations are necessary. Using an inflation rate fixed at a reasonable value ($r = 1.5$), a set of 1000 proteins requires 3 slow loops at the beginning, while at 5000 proteins, the fourth iteration is slow as well. One can expect that 10^5 proteins will need no more than 6-7 slow iterations.
2) Choosing a larger inflation rate reduces the number of slow iterations. However, it is not recommended to use very high inflation rates, because they yield small clusters in the output, which will hardly reveal any biologically relevant protein similarities.

Table I gives us a summary of speed-up ratios reached on input data of various sizes, at different inflation rates. These values were computed against the performance of the conventional TRIBE-MCL algorithm, which computes the whole similarity matrix in every iteration and thus all its iterations last as long as the first loop in our approach. Even higher speed-up ratios are reachable using sparse matrix representation of the similarity matrix, especially in the first iterations, while the majority of nodes in similarity graph are connected to each other. Parallel execution can also improve the efficiency of the algorithm.

TABLE I
SPEED-UP RATIOS REACHED BY THE PROPOSED EFFICIENT EXECUTION SCHEME

Proteins	Inflation rate	Speed-up ratio
1288	1.5	23.02
2106	1.5	21.87
3877	1.5	19.52
6522	1.5	18.26
908	1.3	16.22
908	1.5	25.99
908	1.7	32.77
908	2.0	48.63

V. CONCLUSIONS

In this paper we have proposed an efficient, parallelizable implementation scheme for the graph-based TRIBE MCL clustering method, a useful tool in protein sequence grouping. With this novel formulation, late iterations of the algorithm are performed up to 1000 times quicker, and the overall runtime becomes 15-50 times shorter, than in the conventional case. This speed-up is achieved without any damage of the partition accuracy. Future work will aim at exhaustive tests on larger data sets and the combination of the proposed method with sparse matrix representation and parallel execution, to reduce execution time with another order of magnitude.

REFERENCES

[1] A. J. Enright, S. van Dongen, C. A. Ouzounis, "An efficient algorithm for large-scale detection of protein families", *Nucl. Acids Res.*, vol. 30, pp. 1575–1584, 2002.

[2] M. O. Dayhoff, "The origin and evolution of protein superfamilies", *Fed. Proc.*, vol. 35, pp. 2132–2138, 1976.

[3] H. Hegyi, M. Gerstein, "The relationship between protein structure and function: a comprehensive survey with application to the yeast genome", *J. Mol. Biol.*, vol. 288, pp. 147–164, 1999.

[4] A. Heger, L. Holm, "Towards a covering set of protein family profiles", *Prog. Biophys. Mol. Biol.*, vol. 73, pp. 321–337, 2000.

[5] S. R. Eddy, "Profile hidden Markov models", *Bioinform.*, vol. 14, pp. 755–763, 1998.

[6] S. F. Altschul, T. L. Madden, A. A. Schaffen, J. Zhang, Z. Zhang, W. Miller, D. J. Lipman, "Gapped BLAST and PSI-BLAST: a new generation of protein database search program", *Nucl. Acids Res*, vol. 25, pp. 3389–3402, 1997.

[7] L. Szilágyi, L. Medvés, S. M. Szilágyi, "A modified Markov clustering approach to unsupervised classification of protein sequences", *Neurocomput.*, vol. 73, no. 13-15, pp. 2332–2345, 2010.

[8] A. Andreeva, D. Howorth, J. M. Chadonia, S. E. Brenner, T. J. P. Hubbard, C. Chothia, A. G. Murzin, "Data growth and its impact on the SCOP database: new developments", *Nucl. Acids Res.*, vol. 36, pp. D419–D425, 2008.

[9] T. F. Smith, M. S. Waterman, "Identification of common molecular subsequences", *J. Mol. Biol.*, vol. 147, pp. 195-197, 1981.

[10] S. B. Needleman, C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins", *J. Mol. Biol.*, vol. 48, pp. 443-453, 1970.

[11] Z. Gáspári, K. Vlahovicek, S. Pongor, "Efficient recognition of folds in protein 3D structures by the improved PRIDE algorithm", *Bioinform.*, vol. 21, pp. 3322–3323, 2005.

[12] Protein Classification Benchmark Collection, available at: http://net.icgeb.org/benchmark