Computers in Biology and Medicine 48 (2014) 94–101

Contents lists available at ScienceDirect

Computers in Biology and Medicine

journal homepage: www.elsevier.com/locate/cbm

A fast hierarchical clustering algorithm for large-scale protein sequence data sets

Sándor M. Szilágyi [a,1], László Szilágyi [b,c,*,2]

[a] Petru Maior University, Department of Informatics, Str. Nicolae Iorga Nr. 1, 540088 Tîrgu Mureş, Romania
[b] Budapest University of Technology and Economics, Department of Control Engineering and Information Technology, Magyar tudósok krt. 2, H-1117 Budapest, Hungary
[c] Sapientia University of Transylvania, Faculty of Technical and Human Sciences, Şoseaua Sighişoarei 1/C, 540485 Tîrgu Mureş, Romania

ARTICLE INFO

Article history:
Received 13 November 2013
Accepted 25 February 2014

Keywords:
Protein sequence clustering
Markov clustering
Markov processes
Efficient computing
Sparse matrix

ABSTRACT

TRIBE-MCL is a Markov clustering algorithm that operates on a graph built from pairwise similarity information of the input data. Edge weights stored in the stochastic similarity matrix are alternately fed to the two main operations, inflation and expansion, and are normalized in each main loop to maintain the probabilistic constraint. In this paper we propose an efficient implementation of the TRIBE-MCL clustering algorithm, suitable for fast and accurate grouping of protein sequences. A modified sparse matrix structure is introduced that can efficiently handle most operations of the main loop. Taking advantage of the symmetry of the similarity matrix, a fast matrix squaring formula is also introduced to facilitate the time consuming expansion. The proposed algorithm was tested on protein sequence databases like SCOP95. In terms of efficiency, the proposed solution improves execution speed by two orders of magnitude, compared to recently published efficient solutions, reducing the total runtime well below 1 min in the case of the 11,944 proteins of SCOP95. This improvement in computation time is reached without losing anything from the partition quality. Convergence is generally reached in approximately 50 iterations. The efficient execution enabled us to perform a thorough evaluation of classification results and to formulate recommendations regarding the choice of the algorithm's parameter values.

1. Introduction

Protein families are defined as groups of molecules with relevant sequence similarity [1,2], the members of which are likely to serve similar or identical biological purposes [3]. The identification of protein families is generally performed by clustering algorithms, mostly supported by pairwise similarity or dissimilarity measures [4]. Well established properties of some proteins in a family may be reliably transferred to other members whose functions are not well known [5].

The classification of protein sequences has a wide variety of methods [6], including motif-based classification [7], string kernels [8], data mining techniques [9–11], string weighting [12], word segmentation [13], feature hashing [14], and trees [15]. One of the greatest obstacle for these methods represents the multi-domain structures of many protein families [16]. Several ultra-fast clustering scheme have been introduced recently [17–22] and successfully employed in protein sequence or interaction grouping [23], but also in the clustering of large biochemical and biological networks [24], and documents [25].

TRIBE-MCL is an efficient clustering method based on Markov chain theory [26], introduced by Enright et al [4]. TRIBE-MCL assigns a graph structure to the protein set such a way that each protein has a corresponding node. Edge weights are stored in the so-called similarity matrix S, which acts as a (column or row) stochastic matrix. At any moment, edge weight s_{ij} reflects the posterior probability that protein i and protein j have a common evolutionary ancestor. Initial edge weights in the graph are obtained via pairwise alignment of the sequences, performed by the BLAST search method [27], preferred because of the sparse nature of the provided similarity values.

TRIBE-MCL is an iterative algorithm, performing in each loop two main operations on the similarity matrix: inflation and expansion. Further operations like column or row normalization and matrix symmetrization are included to serve the stability and robustness of the algorithm, and to enforce the probabilistic

* Corresponding author at: Sapientia University of Transylvania, Faculty of Technical and Human Sciences, Şoseaua Sighişoarei 1/C, 540485 Tîrgu Mureş, Romania. Tel.: +40 265 208170; fax: +40 265 206211.
E-mail addresses: szsandor72@yahoo.com (S.M. Szilágyi), lalo@ms.sapientia.ro (L. Szilágyi).
[1] Tel./fax: +40 265 262275.
[2] Tel.: +36 1 463 4027; fax: +36 1 463 2699.

http://dx.doi.org/10.1016/j.compbiomed.2014.02.016
0010-4825

constraint. Inflation raises each element of the similarity matrix to power r, which is a previously established fixed inflation rate. Due to the constraint $r > 1$, inflation favors higher similarity values in the detriment of lower ones. Expansion, performed by raising matrix S to the second power, is aimed to favor longer walks along the graph. Matrix symmetrization has a double role: it maintains the largest value within a column (or row) on the diagonal of the matrix, and eliminates the non-zero similarity values that fall below a previously defined threshold value ε. Clusters are obtained as connected subgraphs in the graph. Convergence reportedly occurs within a few dozens of iterations. TRIBE-MCL is a divisive hierarchical clustering algorithm: once separated, it never connects isolated subgraphs to each other again, it produces the clusters by dividing existing ones into several smaller groups.

This study has the main goal to introduce an efficient implementation of the TRIBE-MCL algorithm using a so-called sparse supermatrix (SSM) data structure, and to investigate the behavior of TRIBE-MCL via thorough tests using the large protein set of the SCOP95 database [28].

The remainder of this paper is structured as follows: Section 2 takes into account the functional details of the TRIBE-MCL algorithm and presents our motivations to introduce modifications. Section 3 presents the details of the proposed efficient TRIBE-MCL algorithm. Section 4 thoroughly evaluates the behavior of the proposed method. Section 5 discusses the achieved results and outlines the role of each parameter, while Section 6 concludes this study.

2. Related works

TRIBE-MCL in its conventional, easily implementable form has a theoretical complexity of $\mathcal{O}(n^3)$, and needs up to 6 h to perform a single loop on a data set containing 10^4 proteins. Although this inefficient solution equally works with any kind of pairwise similarity data, it is prohibitively slow. Accelerating the performance starts with the choice of similarity criterion. For example, BLAST gives us a virtually symmetrical similarity matrix with large amount of zero values. These properties of the matrix are the primary source of acceleration. Since the zero values do not influence any of the TRIBE-MCL operations, it is possible to reduce the computational load by avoiding (skipping) the additions and multiplications with null arguments.

2.1. Matrix splitting

It may occur at any progress level of the algorithm, that a certain column and the corresponding row of S remains with a single non-zero value, which is situated on the diagonal of the matrix. At this point, the node (and protein) corresponding to that certain row can be declared an item isolated in a separate cluster, having no more influence upon the other nodes in the further iterations. Such rows (and columns) can be removed from S, thus reducing the size of the matrix and the computational load of late iterations. This way TRIBE-MCL may be accelerated up to 2 times, depending on input data and parameter values [29]. This idea has recently evolved to the matrix splitting solution, which started from the idea that isolated subgraphs do not get into further interactions with other nodes. This way, when the graph gets torn into isolated subgraphs, which usually occurs after 5–8 iterations, the initial big matrix can be reorganized (via row or column reordering) into several small diagonally situated blocks, and the further computations may neglect the computations with similarities situated outside the blocks, as they are all zero. This way late iterations of TRIBE-MCL can speed-up 10^4 times, and the total runtime becomes 20–50 times shorter [30].

2.2. Sparse matrices

Besides not having to compute additions and multiplications with null arguments, the large amount of zero values in the similarity matrix need not be stored during the TRIBE-MCL iterations. Sparse matrices are suitable data structures for such efficient implementations: while computing the normalization of a column or row, zero elements are not added to the sum, thus reducing the number of additions. In fact, a zero element can only change to non-zero during the expansion. But also in case of the expansion, zero elements in the input do not affect the outcome of any element of the output matrix.

The rest of this section refers to the version of sparse matrix where non-zero elements of columns are stocked in a chained list, ordered by row coordinate. Thus the sparse matrix has an array of list head pointers, each one pointing to the first non-zero element of the corresponding column. Each non-zero element is represented by the structure (row, $value$, $next$). The latter variable in the structure is a pointer to the next non-zero element in the column.

Inflation requires a single parsing of each column and thus the power computation is only performed for non-zero elements. The normalization needs to parse each column twice: first it computes the sum of each column and then it divides all non-zero elements by the sum of the column. Assuring matrix symmetry is more complicated, because it requires searching for the transposed of each non-zero element.

Expansion requires a new sparse matrix for the output. During the computation of the expanded matrix, the elements of each column are determined in such an order, that new non-zero elements are always placed at the end of the list. That is why, it is worth to have a pointer to the tail of the column list as well. Further on, as expansion is computed right after having made the similarity matrix approximately symmetrical, we may find the element s_{ij} as

$$s_{ij}^{(\mathrm{new})} = \sum_{k=1}^{n} s_{ik} s_{kj} \approx \sum_{k=1}^{n} s_{ik} s_{jk} \ , \tag{1}$$

which is easier to compute as columns are way easier to parse than rows in this data structure.

Using such an implementation can reduce the total runtime 100–200 times, but clustering 10^4 proteins still requires a couple of hours [31].

3. Materials and methods

In this paper we introduce an efficient implementation of the TRIBE-MCL algorithm, with the aim of significantly reducing its computational load, without harming the outcome of classification. A special set of data structures will be introduced to facilitate the quick execution of the main operations, and the features of the similarity matrix will be fully exploited to achieve an extremely quick performance. The proposed method will be tested on the proteins of the SCOP95 database.

3.1. The SCOP95 database

The SCOP (Structural Classification of Proteins) database [28] contains protein sequences in order of tens of thousands, hierarchically classified into classes, folds, superfamilies and families [32,33]. The SCOP95 database involved in this study, is a subset of SCOP (version 1.69), which contains $n = 11{,}944$ proteins, exhibiting a maximum similarity of 95% among each other. Pairwise similarity and distance matrices (BLAST [27], Smith–Waterman [34], Needleman–Wunsch [35], PRIDE [36], etc.) are available at the Protein Classification Benchmark Collection [37,38]. In this study