journal homepage: www.intl.elsevierhealth.com/journals/cmpb

A modified two-stage Markov clustering algorithm for large and sparse networks

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

[a] Faculty of Technical and Human Sciences, Sapientia University of Transylvania,Şoseaua Sighişoarei 1/C, 540485 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] Department of Informatics, Petru Maior University, Str. N. Iorga Nr. 1, 540088 Tîrgu Mureş, Romania

A R T I C L E I N F O

Article history:
Received 6 February 2016
Received in revised form
27 May 2016
Accepted 1 July 2016

Keywords:
Hierarchical clustering
Markov clustering
Efficient computing
Sparse matrix
Protein sequence networks

A B S T R A C T

Background: Graph-based hierarchical clustering algorithms become prohibitively costly in both execution time and storage space, as the number of nodes approaches the order of millions.
Objective: A fast and highly memory efficient Markov clustering algorithm is proposed to perform the classification of huge sparse networks using an ordinary personal computer.
Methods: Improvements compared to previous versions are achieved through adequately chosen data structures that facilitate the efficient handling of symmetric sparse matrices. Clustering is performed in two stages: the initial connected network is processed in a sparse matrix until it breaks into isolated, small, and relatively dense subgraphs, which are then processed separately until convergence is obtained. An intelligent stopping criterion is also proposed to quit further processing of a subgraph that tends toward completeness with equal edge weights. The main advantage of this algorithm is that the necessary number of iterations is separately decided for each graph node.
Results: The proposed algorithm was tested using the SCOP95 and large synthetic protein sequence data sets. The validation process revealed that the proposed method can reduce 3–6 times the processing time of huge sequence networks compared to previous Markov clustering solutions, without losing anything from the partition quality.
Conclusions: A one-million-node and one-billion-edge protein sequence network defined by a BLAST similarity matrix can be processed with an upper-class personal computer in 100 minutes. Further improvement in speed is possible via parallel data processing, while the extension toward several million nodes needs intermediary data storage, for example on solid state drives.

* *Corresponding author.* Department of Informatics, Petru Maior University, Str. N. Iorga Nr. 1, 540088 Tîrgu Mureş, Romania. Fax: +40 265 262 275.
Email address: sandor.szilagyi@science.upm.ro (S.M. Szilágyi).

http://dx.doi.org/10.1016/j.cmpb.2016.07.007
0169-2607/© 2016 Elsevier Ireland Ltd. All rights reserved.

1. Introduction

Clustering algorithms have the main goal to structure a certain set of object data into groups based on similarity or dissimilarity criteria [1]. Hierarchical clustering techniques organize the object data into a hierarchy of groups, using either the agglomerative or divisive approach performed on the graph associated to the input data. The agglomerative approach initially considers each item a separate cluster, and gradually merges those clusters that are most similar or least dissimilar at that moment. On the other hand, the divisive approach starts with a single cluster containing all items and builds the hierarchy via recursive splits. An important question in both approaches is where the processing should be stopped, or in other words, which is the ideal number of clusters [2].

As the size of the input data grows, both runtime and memory efficiency are becoming key issues. Several hierarchical clustering solutions have been proposed to address these problems. Gagolewski et al. [3] proposed and validated a modified single link criterion for agglomerative clustering, involving constraints based on cluster size inequality. Loewenstein et al. [4] developed an accurate (non-approximative) agglomerative clustering algorithm upon the centroid-based approach (UPGMA) that does not need all edges loaded at the same time in the memory. The divisive approach performs a more complex job by default, because in any moment there are many more possibilities in choosing a cluster to split and a way of splitting it, than in selecting two clusters to merge. This difficulty is bridged by the Markov clustering (MCL) algorithm [5], as it does not choose a single cluster to split in each iteration. It rather establishes two operations that work against each other: inflation favors strong edges in the graph at the detriment of weak ones, while expansion favors longer walks along the graph. Edges weighted below a predefined threshold are eliminated, so the splitting of clusters is caused in each iteration by vanishing edges. Since Enright et al. [6] demonstrated its accuracy in the identification of protein families, Markov clustering has become a very popular tool in the analysis of protein sequence and interaction networks [7–10], video processing [11], image processing [12], language modeling [13], community detection [14], human action categorization [15], document clustering [16], and FPGA circuit design [17].

Handling graphs with 10^5–10^7 nodes and their corresponding similarity matrices is prohibitively costly in both runtime and storage space. Some of our recent MCL solutions efficiently handled both problems to a certain extent. The best approach to storage limitations employs sparse matrix representations [18,19], while the so-called matrix splitting version of MCL [20] proved runtime efficient during the late iterations of the execution. In this paper, we propose a two-stage solution that optimizes the total runtime of the MCL algorithm running on large (but still loadable) data sets, by combining the two above mentioned solutions. The novel algorithm employs the sparse matrix version during the first few loops (usually 3 to 10, depending on the nature of input data, and the values of inflation rate r and similarity threshold ε), and switching to the matrix splitting version as soon as it becomes favorable. The optimal switching time is automatically detected separately for each isolated subgraph.

The MCL algorithm proposed in this paper differs from the original one [5,6] in the following terms:

1. The proposed algorithm operates on symmetrical graph and matrix, symmetry is enforced in each iteration.
2. Nonzero similarity values on the diagonal of the similarity matrix are never eliminated. This way each row and column contains at least one nonzero similarity value at any moment, which was not the case in Ref. [5]. This makes our hierarchy of clusters easier to interpret.
3. The proposed algorithm obtains clusters as isolated subgraphs of the similarity graph, which are never overlapping each other.

The remainder of this paper is structured as follows. Section 2 reviews some previous MCL versions, whose elements will serve as bricks while building the new solution. Section 3 presents the new aspects of the proposed efficient, two-stage MCL algorithm. Section 4 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. Background

MCL in its conventional (or naive) and easily implementable form has a theoretical complexity of $O(n^3)$, and needs several hours to perform a single loop on a graph containing 10^4 nodes [20]. Although this inefficient solution equally works with any kind of pairwise similarity data, it is prohibitively slow. Accelerating the performance starts with choosing the appropriate similarity criterion. For example, in case of protein sequence data, BLAST gives us a virtually symmetrical similarity matrix with large number of zero values. These properties of the matrix are the primary source of optimization. Runtime efficiency can be achieved via avoiding or skipping additions and multiplications with null arguments, while memory efficiency is possible through avoiding the unnecessary storage of such values.

When a large BLAST similarity matrix is fed to MCL, the corresponding graph usually has all its nodes connected together, but the matrix density is somewhere between 0.1% and 0.5% [21]. After performing a few initial loops, the large graph is divided into several smaller isolated subgraphs. The corresponding matrix of such a graph, after a suitably chosen permutation of rows and columns, contains diagonally placed dense blocks of nonzero values, and zeros outside these blocks. Each diagonally placed block corresponds to an isolated subgraph in the graph. While the initial matrix structure can be more efficiently handled using a sparse matrix representation, these diagonally placed blocks are easier to address within classical two-dimensional arrays. In the following, we will revisit some previous solutions employing both formats.

2.1. *Matrix splitting*

It may occur at any moment during the execution of the algorithm that a certain column and the corresponding row of