A Modified Markov Clustering Approach for Protein Sequence Clustering

Lehel Medvés[1], László Szilágyi[1,2], and Sándor M. Szilágyi[1]

[1] Sapientia - Hungarian Science University of Transylvania,
Faculty of Technical and Human Science, Târgu-Mureş, Romania
medveslehel@yahoo.com, {lalo,szs}@ms.sapientia.ro
[2] Budapest University of Technology and Economics, Department of Control
Engineering and Information Technology, Budapest, Hungary

Abstract. In this paper we propose a modified Markov clustering algorithm for efficient clustering of large protein sequence databases, based on previously evaluated sequence similarity criteria. The proposed alteration consists in an exponentially decreasing inflation rate, which aims at helping the quick creation of the hard structure of clusters by using a strong inflation in the beginning, and at producing fine partitions with a weaker inflation thereafter. The algorithm, which was tested and validated using the whole SCOP95 database, or randomly selected 10-50% sections, generally converges within 12-14 iteration cycles and provides clusters of high quality. Furthermore, a novel generalized formula is given for the inflation operation, and an efficient matrix symmetrization technique is presented, in order to improve the partition quality with relatively low amount of extra computations. A large graph layout technique is also employed for the efficient visualization of the obtained clusters.

Keywords: Markov clustering, protein sequence clustering, sparse matrix, large graph layout, SCOP95 database.

1 Introduction

One of the main goals of functional genomics is to establish protein families in large databases. Successful classification of protein families can have significant contributions to the delineation of functional diversity of homologous proteins, and can provide valuable evolutionary insights as well [9].

By definition, protein families represent groups of molecules showing relevant sequence similarity [4]. Members of such protein families may serve similar or identical biological purposes [12]. Identifying these families is generally performed by clustering algorithms, which are supported by similarities and/or dissimilarities, previously computed between all pairs of protein sequences.

Performing an accurate clustering results in such protein families, whose members are related by a common evolutionary history [10]. If this condition holds, well established properties of some proteins in the family may be reliably transferred to other members whose functions are not well known [11]. Several protein clustering methods are currently available in the literature [7,14]. One of

M. Chetty, A. Ngom, and S. Ahmad (Eds.): PRIB 2008, LNBI 5265, pp. 110–120, 2008.

the greatest obstacle for them represents the multi-domain structures of many protein families [5].

TRIBE-MCL is an efficient protein sequence clustering method proposed by Enright et al. [9], based on Markov chain theory [6,7]. The authors assigned a graph structure to the protein database such a way that each protein has a corresponding node, while edge weights in the graph represent a priori computed similarity values, obtained via BLAST search methods [2]. Clusters were then obtained by alternately applying two operations to the similarity matrix: inflation and expansion. The former represents a task, which re-evaluates the values within columns of the matrix by raising higher probabilities and suppressing the small ones, while the latter aims at favoring longer walks along the graph, which is obtained via matrix squaring.

In this paper we propose a modification of the TRIBE-MCL algorithm, in order to enhance its accuracy and improve its time complexity. This is achieved by introducing a time-varying inflation rate and thus forcing the algorithm to apply a stronger inflation in the first iteration when the hard structure of the clusters is established, and reduce inflation strength for fine tuning the cluster shape in later iterations.

The remainder of this paper is structured as follows: Section 2 takes into account the functional details of the TRIBE-MCL algorithm and presents the proposed modifications. Section 3 presents our own considerations upon large graph layout techniques. Section 4 evaluates and discusses the efficiency and accuracy of the proposed method. Section 5 presents the conclusions and gives some hints for further research.

2 The Proposed Markov Clustering Approach

2.1 The TRIBE-MCL Algorithm

TRIBE-MCL is an iterative algorithm, which operates on a directional graph. The nodes of the graph represent the protein sequences we wish to cluster, while edges show the similarity between pairs of protein sequences. The edge lengths are stored in a so called similarity matrix. Theoretically, the initial edge lengths can be computed using any sequence alignment method. However, the convergence speed will depend on the initial similarities.

In most of the cases, this initial similarity matrix is not symmetrical. This may be treated as a problem or not. If one only wishes to cluster the sequences, that is, to find certain groups of proteins, which show high similarity within the cluster and low similarity between different clusters, then using symmetrical similarity matrix is recommendable. Asymmetrical similarity values, however, may reveal the direction of evolution among the proteins situated within a given cluster. Consequently, making the similarity matrix symmetrical in every iteration is an extra step, whose benefits and costs will be tested in the followings.

The TRIBE-MCL algorithm treats the similarity matrix as a stochastic matrix [8]. In such a matrix, probability or possibility values are stored: in general,

S_{ij} represents the possibility, that protein i becomes protein j during an evolutionary step. A decision has to be made at the beginning, whether S is treated as a column stochastic matrix or a row stochastic matrix. The difference is significant: if S is a column stochastic matrix, the columns are normalized in every iteration and thus they become probability values of past evolutionary steps: for example S_{ij} shows what is the probability that protein j was i before the latest evolutionary step. On the other hand, rows of a column stochastic matrix are not normalized, so values in row j show the possible outcomes of a next evolutionary step, and their likelihood values, which are not probabilities as their sum is not 1. In this paper we chose to treat the similarity matrix as a column stochastic matrix.

The TRIBE-MCL algorithm consists of two main operations, namely the inflation and expansion, which are repeated alternately until a convergence is reached, that is, clusters become stable. Inflation has the main goal to favor more likely direct walks along the graph in the detriment of less likely walks, while expansion reveals possible longer walks along the graph.

2.2 The Inflation Operation

The inflation operation has the main goal to modify the similarity values within the columns of the similarity matrix such a way, that differences gain some emphasis. In other words, inflation favors more probable walks over less probable walks along the protein graph [9]. Literature recommends using the following inflation operation:

$$S_{ik}^{(n+1)} = \frac{\left(S_{ik}^{(n)}\right)^r}{\sum_{j=1}^{N}\left(S_{jk}^{(n)}\right)^r} \quad , \tag{1}$$

where r represents the inflation rate, which controls the strength of inflation. The larger the inflation rate, the more favored will be the high similarities.

Besides the inflation itself, an intentional side effect is the normalization of the columns: whatever the similarity values were before inflation (except for a zero column, which is unlikely to occur), the column will sum up at one after the operation.

If we examine the evolution of the number of clusters of different sizes (one such representation can be seen in Fig. 2(left)), we can remark, that TRIBE-MCL has two stages of its runtime: it needs some iterations until the changes influence the number of clusters significantly. The end of this first stage is shown by a significant maximum in the numbers or small clusters (not singletons). The number of iterations in this first stage strongly depends on the inflation rate: in case of $r = 1.1$, this first stage may need 7-8 iterations, while in case of $r = 1.5$, 3-4 iterations suffice. During the second stage, the number of small clusters decreases and finally stabilizes after 8-12 iterations.

In our opinion, the inflation rate has to be chosen such a way, that it is large enough in the first stage, so that the formation of cluster begins quickly and thus

the first stage doesn't last too long. On the other hand, in the second stage the inflation rate has to be small enough, to obtain high-quality clusters. In order to deal with both these requirements, in this paper we propose the usage of a variable inflation rate, given by the equation:

$$r^{(n)} = 1 + r_0 \times \exp\left(-\frac{n}{\tau}\right) \ , \tag{2}$$

where $1 + r_0$ represents the initial inflation rate, n is the ordinal number of the current iteration, and τ is a time constant. In this paper we use: $r_0 = 2$ and $\tau = 10$, which gives the inflation rate the variation shown in Fig. 3(right).

This inflation operation could be generalized in the following manner: let us define a continuous function $I : [0, +\infty) \to [0, +\infty)$, $I(0) = 0$, $I'(x) > 0$, $I''(x) > 0 \ \forall x > 0$. It can be proved, that the generalized inflation, defined as:

$$S_{ik}^{(n+1)} = \frac{I(S_{ik}^{(n)})}{\sum_{j=1}^{N} I(S_{jk}^{(n)})} \ , \tag{3}$$

where I was established according to the above mentioned conditions, favors high similarities over low ones. Nevertheless, this generalized formula give us a higher freedom to choose the inflation operation. Obviously, setting $I(x) = x^r$ returns us to (1).

2.3 The Expansion Operation

The expansion operation is associated with random walks of higher lengths along the graph, which may include several steps [9]. It is computed with the normal matrix squaring operation. It produces new probabilities with all pairs of nodes, where one node is the point of departure and the other is the destination. Obviously, we will get high probabilities for pairs of nodes situated within the same cluster, and low ones for nodes from different clusters.

Expansion needs two instances of the similarity matrix. Consequently this is the operation, which determines the amount of directly processable protein sequences, due to the memory limitations of the PC.

2.4 Matrix Symmetry

Even if similarity measures are initially symmetrized, this property gets lost after the first inflation, due to the nature of the operator, as it treats the similarities as a column stochastic matrix. If there is any reason (dictated by the biological scenario) for which symmetrical similarity matrix is required, the following extra processing step should be inserted in every iteration of the TRIBE-MCL.

1. For any $i, j = 1 \ldots n$, $i \neq j$, if $|S_{ij} - S_{ji}| > \varepsilon$, set $S_{ij}^{(\text{new})} = S_{ji}^{(\text{new})} = \sqrt{S_{ij} \times S_{ji}}$, where ε is a previously set small constant.
2. Normalize the columns of the stochastic matrix.
3. Repeat steps 1-2, until symmetry is reached with ε tolerance.

2.5 Implementation Issues

The amount of protein sequences involved in clustering is theoretically limited by the following relation:

$$2 \times N^2 \times d \leq M \ , \tag{4}$$

where d represents the memory required by one probability value (acceptable resolution requires 16 bits), and M is the available amount of memory. Considering 1GB storage space, this means a theoretical limit of $N < 16384$ proteins. Most protein databases contain even more data. When the necessary storage for two similarity matrices required by the expansion is not available, we propose using a sparse matrix representation. If we suppose three decimal representation of transition probabilities, a maximum number of 1000 values can be nonzero in each column, but practically their count will be less with at least an order of magnitude. So the sparse matrix representation, even if needs three times more space for a single probability value, can reduce the necessary memory, and can significantly increase the amount of simultaneously processable proteins.

2.6 Algorithm

The proposed algorithm can be summarized as follows:

1. Compute initial similarity matrix.
2. Inflate the similarity data according to Eq. (3).
3. Expand the similarity data via matrix squaring.
4. Symmetrize the similarity matrix if desired.
5. Repeat steps 2-4 until transition probabilities stabilize. Generally 10-15 iterations suffice.

3 Graph Visualization

By applying the proposed Markov clustering method to subsets of the SCOP95 database, we obtain a few large clusters among the small ones. In order to visualize the structure of such large clusters, we propose a modified version of the large graph layout (LGL) algorithm given in [1].

The subgraph obtained from the Markov clustering (from now on: subgraph), representing a large cluster, provides the input data for the proposed layout generating algorithm.

The proposed algorithm places the nodes within the setting gradually, and allows them to move according to some attractive and repulsive forces that interact among them (see Fig. 1). The magnitude of the attractive forces between two given nodes only depends on their similarity value, their relative position only influences the direction of the force. There is also a repulsive force between any two nodes, whose strength only depends on their physical distance. This force has the main goal to keep nodes distant from each other. Short distance implies extremely strong repulsive force, which loses its strength if distances grow, and at a given distance limit the repulsive force is extinguished.

These forces are not meant in physical terms, and the equation (8) that describes the movement of nodes doesn't correspond to physical laws either.

The computation of the graph is performed according to the following rules:

1. As a first step, a minimum spanning tree (MST) of the subgraph is generated, using as weights the dissimilarity values obtained as a negative power of the computed similarities. This MST will establish the order in which the nodes will be placed into the layout.
2. One of the nodes in the MST needs to be declared central node. This node could be arbitrarily chosen, however, we always choose the node showing largest similarities to its neighbors.
3. The central node is placed into the origin, and it's frozen in that position.
4. We look for the neighbors of the central node in the MST, and place them on a hypersphere having its center in the origin and its radius of unit length. We let these nodes move without leaving the surface of the hypersphere, according to the attractive and repulsive forces that interact among them. Finally these nodes are frozen in their stabilized positions.
5. We look for the neighbors of the nodes found in the previous step in the MST, and place them on a double-radius hypersphere according to the following formula:

$$\boldsymbol{v}_{\text{new node}} = \left(\frac{\boldsymbol{v}_{\Omega}}{||\boldsymbol{v}_{\Omega}||} + \frac{\boldsymbol{v}_{\text{parent}} - \boldsymbol{v}_{\text{grandparent}}}{||\boldsymbol{v}_{\text{parent}} - \boldsymbol{v}_{\text{grandparent}}||} \right) + \boldsymbol{v}_{\text{parent}} + \nu \ , \qquad (5)$$

where the notation $\boldsymbol{v}$ refers to position vector, Ω represents the set of nodes already present in the layout, and ν is a random noise vector, which assures that newly introduced nodes will not be placed into identical positions. The movement of nodes are governed by attractive and repulsive forces among them. The sum of forces that influence node n_k is given by

$$\boldsymbol{F}_k = \sum_{i \in \Omega_k} \left[\boldsymbol{F}_{i,k}^{(\text{a})} + \boldsymbol{F}_{i,k}^{(\text{r})} \right] \ . \qquad (6)$$

where $\boldsymbol{F}_{i,k}^{(\text{a})}$ stands for the attractive force emerging from the similarity between proteins represented by nodes n_i and n_k, while $\boldsymbol{F}_{i,k}^{(\text{r})}$ is the repulsive force born from the mutual positions of nodes n_i and n_k, and $\Omega_k = \Omega - \{k\}$. The forces influencing node n_k are computed as follows:

$$\boldsymbol{F}_k = \sum_{i \in \Omega_k} \left[\frac{(\boldsymbol{v}_i - \boldsymbol{v}_k)}{||\boldsymbol{v}_i - \boldsymbol{v}_k||} \times S(n_i, n_k) + (\boldsymbol{v}_i - \boldsymbol{v}_k) \times g\left(||\boldsymbol{v}_i - \boldsymbol{v}_k||\right) \right] \ , \qquad (7)$$

where $S(n_i, n_k) = S_{ik}$ is the similarity value provided by the Markov clustering, and $g(\cdot)$ represents the function that describes the behavior of repulsive forces: it is considered as an exponentially decreasing function that reaches the zero value at a given distance. The movement of node n_k is described by the equation

$$\boldsymbol{v}_k \leftarrow \boldsymbol{v}_k + \Delta t \times \boldsymbol{F}_k \ , \qquad (8)$$

where Δt is the considered time step. As a final computation step, the position vector v_k is brought back to its hypersphere of radius r_k:

$$\boldsymbol{v}_k \leftarrow \boldsymbol{v}_k \times \frac{r_k}{||\boldsymbol{v}_k||} \quad . \tag{9}$$

Nodes are frozen in the stable position they reach in several movement steps. A single movement step is depicted in Fig. 1.

6. Repeat the previous step until all nodes of the subgraph will be included into the layout.

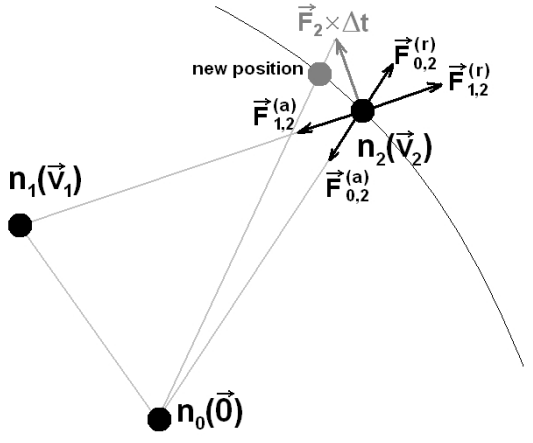

Fig. 1. Balance of forces of one node in case of a 3-node setting

4 Results and Discussion

The proposed modified TRIBE-MCL method was evaluated using the SCOP95 database [3,13], which contains protein sequences that show at most 95% similarity with each other.

The number of protein sequences in the database is quite large to handle using a PC. That's why, in some cases, we decided to randomly choose only a part (multiples of 10%) of the database to test the efficiency and accuracy of the clustering.

Tests revealed the efficiency of the proposed algorithm: the first stage, during which the hard structure is established, usually requires 3 or 4 iteration cycles. The second stage needs further 8-10 cycles to reach convergence.

Figures 2-4 show the results of the algorithm performed on 40% of the SCOP95 database. Figure 2 (left) shows the varying number of clusters of different sizes over 20 iterations.

The boundary between the running stages of the algorithm, indicated by the maxima, are clearly visible at iteration number 3. Figure 2 (right) indicates the total number of non-singleton cluster after each cycle. This latter image indicates

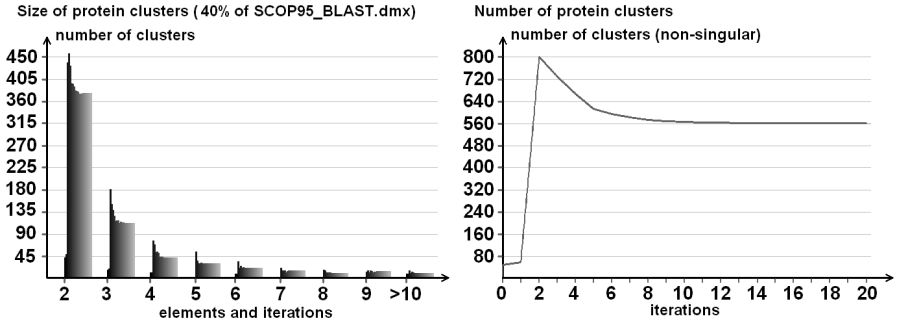

Fig. 2. (left) The evolution of number of different sized clusters, during 20 iteration cycles. The first stage needed four iterations, while the full convergence required an 8-cycle second stage; (right) Graphical representation of the number of non-singleton clusters vs. iteration index.

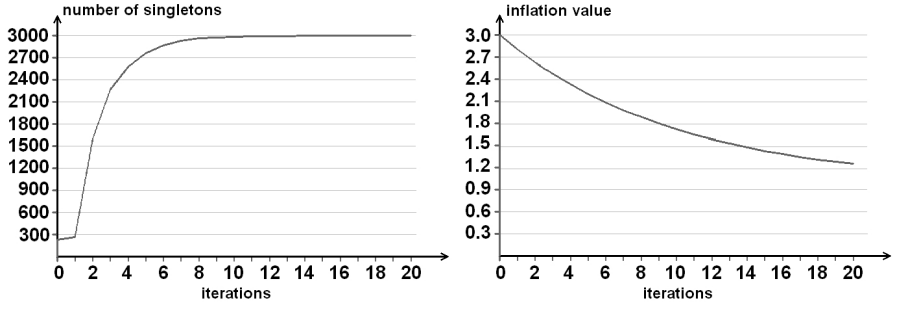

Fig. 3. (left) The first rising, then converging number of singletons; (right) The variable inflation rate, represented vs. iteration count

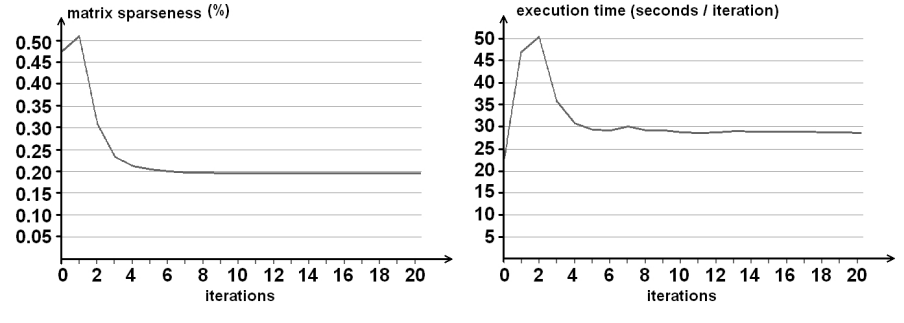

Fig. 4. The varying inner parameters of the algorithm: the sparseness of the similarity matrix (left) and the duration in time of each iteration, when the input data was randomly chosen 40% of SCOP95 database

Fig. 5. The best sequence alignment found in the largest cluster of randomly chosen 40% of the SCOP95 proteins

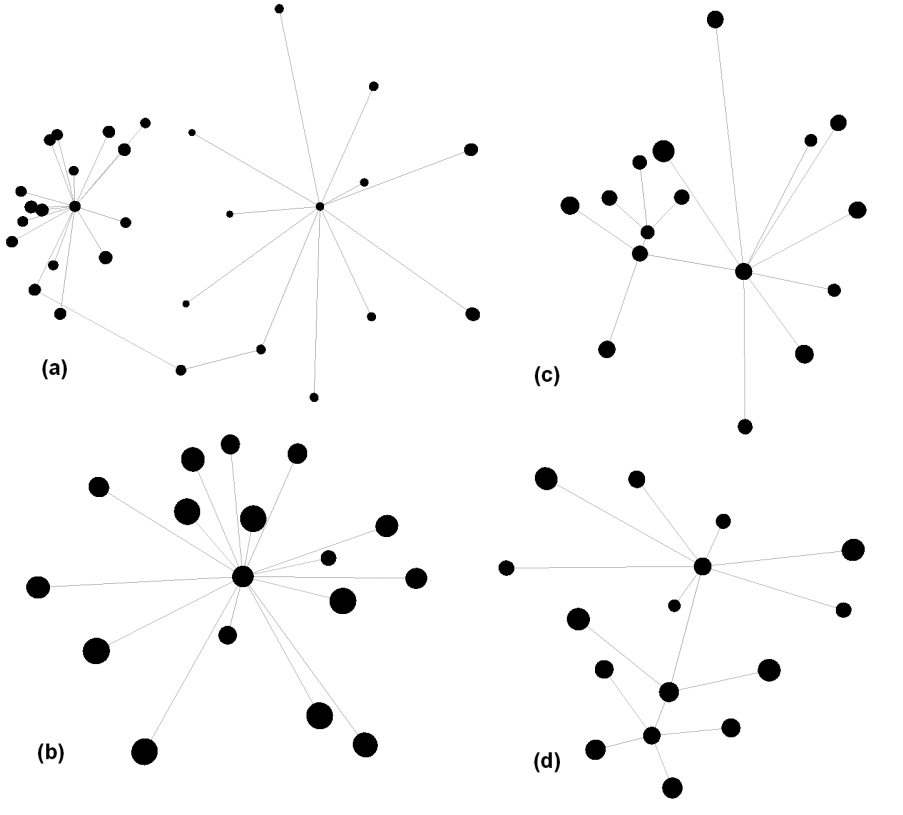

Fig. 6. 3D representation of a given cluster during the iterations of the modified TRIBE-MCL algorithm: (a) after 2 iterations, (b) after 5 iterations, (c) after 8 iterations, (d) after 10 iterations

iteration 14 to be the one, when convergence is established. Figure 3 (left) shows the variation of the number of detected isolated protein sequences in the input data. Figure 3 (right) is a graphical representation of the variable inflation rate, using the parameters proposed in the previous chapter.

Figure 4 shows the runtime parameters: on the left side we can see the sparseness of the similarity matrix, when we used four decimals resolution for similarity values. Sparseness values below 0.5% means that only one out of 200 probabilities is non-zero during the computations, so the number of simultaneously processable proteins can increase 8-10 times, if we turn to sparse matrix representation within the bounds of the the limited storage space. Figure 4 (right) indicates the time necessary to process each iteration with a PC having Athlon64 3200+ processor and 1GB RAM.

Figure 5 shows the alignment of those two protein sequences, which were found the most similar ones inside the largest obtained cluster.

Figure 6 presents the aspect of a given cluster at different stages of the Markov clustering. After two iterations, the graph still consists of a weak union of two clusters. After five iterations, the set of proteins that would finally belong to the cluster is mostly established (only one node will leave the cluster after this point). But the structure of the cluster still undergoes several slight changes, as shown in the representations of later stages.

Finally, Fig. 7 shows a 2D graph representation of the largest cluster found, produced with the proposed large graph layout algorithm.

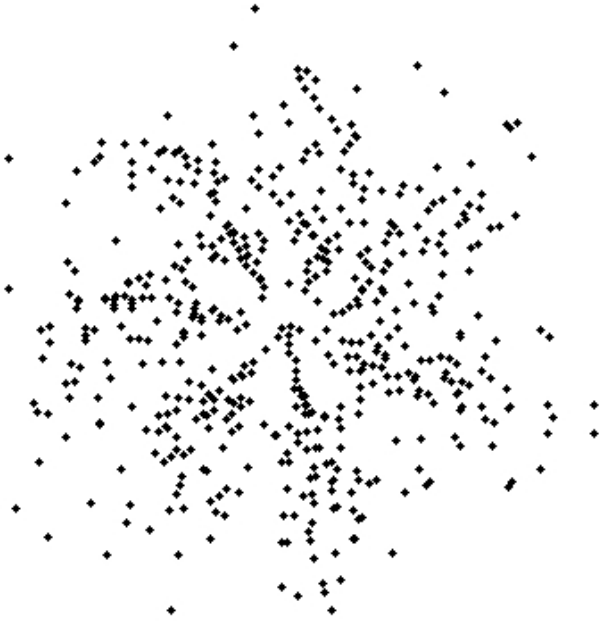

Fig. 7. A 2D graph representation of the largest cluster, consisting of 542 proteins

5 Conclusions

In this paper we proposed a modification in the TRIBE-MCL algorithm, in the terms of a variable inflation rate. An exponentially decreasing value of the inflation rate was recommended in order to deal with the nature of the problem: a high inflation rate at the beginning serves the quick establishment of hard structure of clusters, while a lower inflation rate in the followings serves the final partition quality. The proposed algorithm was found efficient in time and accurate in forming protein families. We also proposed a similarity matrix symmetrization scheme, for the case when clustering intends to ignore the evolutionary direction. Moreover, we also presented a general formulation to the inflation operation, giving the theoretical conditions of the inflation function that can replace the simple power function. Future works will aim at implementing and testing this latter proposal, and at enhancing the LGL algorithm to provide finer representation of the obtained clusters.

References

1. Adai, A.T., Date, S.V., Wieland, S., Marcotte, E.M.: LGL: Creating a map of protein function with an algorithm for visualizing very large biological networks. J. Mol. Biol. 340, 179–190 (2004)
2. Altschul, S.F., Madden, T.L., Schaffen, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped BLAST and PSI-BLAST: a new generation of protein database search program. Nucleic Acids Res. 25, 3389–3402 (1997)
3. Andreeva, A., Howorth, D., Brenner, S.E., Hubbard, T.J.P., Chothia, C., Murzin, A.G.: SCOP database in 2004: refinements integrate structure and sequence family data. Nucl. Acids Res. 32, D226–D229 (2004)
4. Dayhoff, M.O.: The origin and evolution of protein superfamilies. Fed. Proc. 35, 2132–2138 (1976)
5. Doolittle, R.F.: The multiplicity of domains in proteins. Ann. Rev. Biochem. 64, 287–314 (1995)
6. Eddy, S.R.: Hidden Markov models. Curr. Opin. Struct. Biol. 6, 361–365 (1996)
7. Eddy, S.R.: Profile hidden Markov models. Bioinf. 14, 755–763 (1998)
8. Enright, A.J., Ouzounis, C.A.: BioLayout: an automatic graph layout algorithm for similarity visualization. Bioinf. 17, 853–854 (2001)
9. Enright, A.J., van Dongen, S., Ouzounis, C.A.: An efficient algorithm for large-scale detection of protein families. Nucleic Acids Res. 30, 1575–1584 (2002)
10. Fitch, W.M.: Aspects of molecular evolution. Ann. Rev. Genet. 7, 343–380 (1973)
11. Heger, A., Holm, L.: Towards a covering set of protein family profiles. Prog. Biophys. Mol. Biol. 73, 321–337 (2000)
12. Hegyi, H., Gerstein, M.: The relationship between protein structure and function: a comprehensive survey with application to the yeast genome. J. Mol. Biol. 288, 147–164 (1999)
13. Lo Conte, L., Ailey, B., Hubbard, T.J., Brenner, S.E., Murzin, A.G., Chothia, C.: SCOP: a structural classification of protein database. Nucleic Acids Res. 28, 257–259 (2000)
14. Tsoka, S., Ouzounis, C.A.: Recent developments and future directions in computational genomics. FEBS Lett. 480, 42–48 (2000)