J.Thi-Qar Sci. Vol.3 (1) July/2011

ISSN 1991- 8690 الترقيم الدولي ٨٦٩٠ - ١٩٩١

Enhanced MCL Clustering

Kadhem Mahdi Hashem Mouiad Abid Hani

University of Thi Qar-College of Education-Dept. of computer science

Abstract

The goal of graph clustering is to partition vertices in a large graph into different clusters based on various criteria such as vertex connectivity or neighborhood similarity. Graph clustering techniques are very useful for detecting densely connected groups in a large graph. In this research, we introduce a clustering algorithm for graphs; this algorithm is based on Markov clustering (***MCL***), which is a clustering method that uses a simulation of stochastic flow. We have tuned to set the proper factors of inflation, matrix and threshold. Theoretical analysis is provided to show that the enhanced ***EMCL***-Cluster is converging. Then the proposed method is compared with other clustering methods.

Keywords: Markov clustering (***MCL***), Markov Chain Model (***MCM***), Repeated Random Walks (***RRW***), Graph Clustering (***GC***).

1-Introduction

Graph clustering is a useful and important unsupervised learning technique widely studied in many researches [1, 2]. The general goal of clustering is to group similar objects into one cluster while partitioning dissimilar objects into different clusters. Clustering has broad applications including the analysis of business and financial data, biological data, time series data, spatial data, and text understanding [3, 4] The grouping is usually based on some *similarity measure* defined for the data elements, so clustering is strongly related to *unsupervised learning* in pattern recognition systems [5]. Graph as an expressive data structure is popularly used to model structural relationship between objects. Graph clustering is an interesting and challenging research problem which has received much attention recently [4, 5]

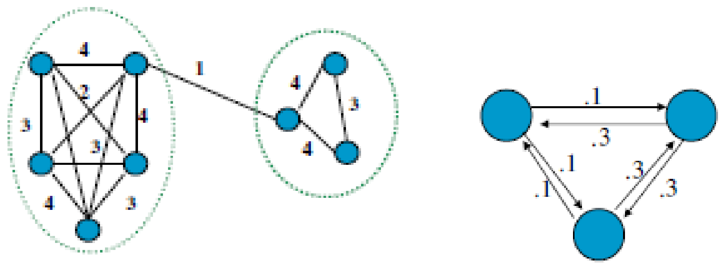
1.1 Graph Notation

Let $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ be an undirected graph with vertex set $\boldsymbol{V} = \{\boldsymbol{v_1}. . . \boldsymbol{v_n}\}$, see figure (1, *a*). In the following we assume that the graph $\boldsymbol{G}$ is weighted, that is each edge between two vertices $\boldsymbol{v_i}$ and

v_j carries a non-negative weight $w_{ij} \geq 0$, see figure (1, *b*). The weighted *adjacency matrix* of the graph is the matrix $W = (w_{ij})_{i,j} = 1,...,n$. If $w_{ij} = 0$ this means that the vertices v_i and v_j are not connected. As G is undirected we require $w_{ij} = w_{ji}$. The degree of a vertex $v_i \in V$ is defined as

$$d_i = \sum_{j=1}^{n} w_{ij}$$

A subset $A \subset V$ of a graph is connected if any two vertices in A can be joined by a path such that all intermediate points also lie in A. A subset A is called a connected component if it is connected and if there are no connections between vertices in A and $\overline{A}$. The sets $A_1 . . . A_k$ form a partition of the graph if $A_i \cap A_j = Ø$; and $A_1 \cup . . . \cup A_k = V$.



(*a*) (*b*)

Figure (1) major kinds of Graph

1.2 Graph-based Clustering Algorithms

A graph-based clustering will first construct a graph or hypergraph and then apply a clustering algorithm to partition the graph or hypergraph. A link-based clustering algorithm can also be considered as a graph clustering, because we can think of the links between data points as links between the graph nodes. There are a number of graph-based clustering algorithms; some of them are listed below.

1.2.1 CACTUS algorithm

CACTUS (**CA**tegorical **C**lus**T**ering **U**sing **S**ummaries) algorithm is developed by Ganti et el. (1999), the interattribute and intra-attribute summaries of the database are constructed and then a graph called the similarity graph is defined according to those summaries. Finally, the clusters are found with respect to those graphs.

The CACTUS algorithm

1. Compute the interattribute and intra-attribute summaries from the database {summarization phase}
2. Analyze each attribute to compute all cluster projections on it, and then synthesize candidate clusters on sets of attributes from the cluster projections on individual attributes {clustering phase}
3. Compute the actual clusters from the candidate clusters {validation phase}

1.2.2 ROCK algorithm

ROCK (**RO**bust **C**lustering using Lin**K**s) algorithm is developed by Guha et al. (2000), and is an agglomerative hierarchical clustering algorithm that employs links to merge clusters. ROCK uses link-based similarity measure to measure the similarity between two data points and between two clusters

```
ROCK Algorithm
Require: $n$: number of data points; $D$: data set;
1:   for $i$ = 1 to $n$-1 do
2:     for $j$ = i+1 to $n$ do
3:       Compute $link(p_i,p_j)$
4:     end for
5:   end for
6:   for $i$ = 1 to $n$ do
7:     build a local heap $q[i]$ that contains every cluster $C_j$ such that $link(C_i,C_j)$ is nonzero
           {at first, each data points forms a cluster}
8:   end for
9:  Build a global heap Q that is ordered in decreasing order of the best goodness
    measure and contains all the clusters
10:  repeat
```

2-Related Works

In his Ph.D. thesis, Van Dongen [9, 2000] has presented the theoretical and practical applications of Markov Clustering (***MCL***) algorithm. Chris Biemann [12, 2005] has used a different modification of ***MCL*** in Natural Language Processing (NLP) applications. H. S. Shon et al [13, 2007] proposed a clustering approach using ***MCL*** algorithm for analyzing microarray data, and they provided a detailed explanation of their new clustering algorithm; Chinese Whispers; which uses the only the strong points in Markov clustering algorithm. The Chinese Whispers is given below:

```
Initialize:
For all $v_i$ in V: class($v_i$)=i;
While changes:
For all $v_i$ in V, randomized order:
Class(v)=highest ranked class
in neighborhood of v;
End
```

In [5, 2007], In this survey an overview the definitions and methods for graph clustering are given, that is, finding sets of “related” vertices in graphs. They review the many definitions for what is a cluster in a graph and measures of cluster quality. Then they present global algorithms for producing a clustering for the entire vertex set of an input graph, after which they discuss the task of identifying a cluster for a specific seed vertex by local computation.

3-Markov chains and Random walk

A *Markov chain* is a stochastic process in which future states only depend on the current state, not the past, taking values from some countable state space. The probabilities for moving to another state from current state form the *transition matrix* of the Markov chain. In general, each Markov chain, independent of how the transition probabilities are defined, can be represented by a weighted directed graph where each state corresponds to
a vertex, each edge corresponds to a transition that has nonzero probability and the edge weight is the probability in question [5]. For an unweighted graph, when one moves from one vertex to another choosing a neighboring vertex uniformly at random, the transition matrix that results is the *normalized* adjacency matrix $\boldsymbol{D}(\boldsymbol{G})^{-1}\boldsymbol{A}(\boldsymbol{G})$ of the graph $\boldsymbol{G}$. This means that the probability of moving from vertex v to w is simply:

$$p_{v,w} = \begin{cases} \frac{1}{\deg \mathrm{v}} \text{if } \mathrm{w} \in \mathrm{V} \\ 0 \text{ otherwise,} \end{cases}$$

Such a walk is called *random*, *blind*, *regular* or *simple*, as it is but one of many possible definitions of walks in graphs. The *first passage time* from state ***j*** to state ***i*** is the time step when the chain first visits state ***i*** when started at state ***j***. The *absorption time* from state ***j*** to state ***i*** is the first passage time in a *modified* chain, where state ***i*** is made into an absorbing state by removing all of its outbound transitions. The spectrum of the transition matrix can be used to evaluate the *mixing time* of the chain, which is the time it takes for the chain to reach its *stationary distribution*. The stationary distribution is a distribution that no longer changes over time as more and more transitions are being performed. It defines for each state the probability that the walk is at that state if a single observation is made after the walk has been run for a sufficiently long time. The stationary distribution can be obtained by computing the *left* eigenvector corresponding to the *largest* eigenvalue of the transition matrix [6]. ***MCL*** is based on Markov chain model both in theory and practice [8]. The transition matrix of the Markov chain could be converted into a proper column-stochastic matrix; that is, a matrix with nonnegative elements (since they values represent probabilities), and the elements of each column sum up 1, see figure (2).

$$\begin{pmatrix} 0 & .25 & .33 & .33 & 0 & 0 & 0 \\ .33 & 0 & .33 & .33 & .33 & 0 & 0 \\ .33 & .25 & 0 & .33 & 0 & 0 & 0 \\ .33 & .25 & .33 & 0 & 0 & 0 & 0 \\ 0 & .25 & 0 & 0 & 0 & .5 & .5 \\ 0 & 0 & 0 & 0 & .33 & 0 & .5 \\ 0 & 0 & 0 & 0 & .33 & .5 & 0 \end{pmatrix}$$

Figure (2) Column-stochastic matrix

4-The Proposed Method

4.1-Enhanced Markov Clustering (MCL) Algorithm

The ***MCL*** employs the normalized adjacency matrix of ***G***, i.e. $\boldsymbol{M}(\boldsymbol{G}) = \boldsymbol{D}(\boldsymbol{G})^{-1}\boldsymbol{A}(\boldsymbol{G})$ where ***A*** (***G***) is the adjacency matrix and ***D***(***G***) the diagonal matrix of vertex degrees. The idea behind *Markov Clustering* (***MCL***) is that a "random walk that visits a dense cluster will likely not leave the cluster until many of its vertices have been visited." Rather than actually simulating random walks, ***MCL*** iteratively modifies a matrix of transition probabilities. Starting from $\boldsymbol{M} = \boldsymbol{M}(\boldsymbol{G})$ (which corresponds to random walks of length at most one), the following two operations are iteratively applied:

1. *Expansion*, in which ***M*** is taken to the power $\boldsymbol{e} \in \boldsymbol{N} > 1$ thus simulating ***e*** steps of a random walk with the current transition matrix (Algorithm 1, Step 1)
2. *Inflation*, in which ***M*** is re-normalized after taking every entry to its r^{th} power, $\boldsymbol{r} \in \boldsymbol{R}^{+}$. (***EMCL*** Algorithm, Steps 2–4).

Note that for $\boldsymbol{r} > 1$, inflation emphasizes the heterogeneity of probabilities within a row, while for $\boldsymbol{r} < 1$, homogeneity is emphasized. The iteration is halted upon reaching a recurrent state or a fixpoint. A recurrent state of period $\boldsymbol{k} \in \boldsymbol{N}$ is a matrix that is invariant under ***k*** expansions and inflations, and a fixpoint is a recurrent state of period 1. It is argued that ***EMCL*** is most likely to end up in a fixpoint. The clustering is induced by connected components of the graph underlying the final matrix. Pseudo-code for ***EMCL*** is given in Algorithm. Except for the stop criterion, ***EMCL*** is deterministic, and its complexity is dominated by the expansion operation which essentially consists of matrix multiplication. The enhanced ***MCL*** clustering algorithm is running in approximate steps of 88% compared to the steps of the ordinary ***MCL*** clustering algorithm. This is achieved by running the ordinary ***MCL*** clustering in 100 steps and compares the results of each step with all the steps obtained by running the enhanced ***MCL***. We found that the results reached in step 44 when implementing ordinary ***MCL*** clustering are reached in step 40 when implementing enhanced ***MCL***. The reason of this enhancement is the difference matrix ***DM*** use which leads to the decrease in number of steps required to reach the same results (steps 4.1 and 4.2).

Algorithm: Enhanced Markov Clustering (***EMCL***)

Input: $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, expansion parameter ***e***, inflation parameter ***r***, empty matrix called difference matrix ***DM***.

```
Begin
    $M \leftarrow M(G)$
    While $M$ is not fixpoint do
1       $M \leftarrow M^e$
2       for all $u \in V$ do
3           for all $v \in V$ do $M_{uv} \leftarrow M^r_{uv}$
4.1             $DM = abs[(M_{uv})_i - (M_{uv})_{i-1}]$
4.2             If $(DM)$ > threshold then
                $(M_{uv})_i = (M_{uv})_i + DM$
4           for all $v \in V$ do $M_{uv} \leftarrow \frac{M^r_{uv}}{\sum_{w \in V} M_{uw}}$
```

H ← graph induced by non-zero entries of ***M***
C ← clustering induced by connected components of ***H***
End
The fixpoint mentioned in the algorithm represents the recurrent state that the matrix should reach after a finite number of steps .

4.2-Results

We have implemented ***EMCL*** algorithm in Delphi language and we set the proper factors of inflation, matrix and threshold. The inflation factor that we have chose concisely,

$$\begin{pmatrix}
0.2 & 0.25 & 0 & 0 & 0 & .333 & .25 & 0 & 0 & .25 & 0 & 0 \\
0.2 & 0.25 & .25 & 0 & .2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.25 & .25 & .20 & .20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 25 & .20 & 0 & 0 & 0 & .20 & .20 & 0 & .20 & 0 \\
0 & 0.25 & .25 & 0 & .20 & 0 & .25 & .20 & 0 & 0 & 0 & 0 \\
0.2 & 0 & 0 & 0 & 0 & .333 & 0 & 0 & 0 & .25 & 0 & 0 \\
0.2 & 0 & 0 & 0 & .20 & 0 & .25 & 0 & 0 & .25 & 0 & 0 \\
0 & 0 & 0 & .20 & .20 & 0 & 0 & .20 & .20 & 0 & .20 & 0 \\
0 & 0 & 0 & .20 & 0 & 0 & 0 & .20 & .20 & 0 & .20 & .333 \\
0.2 & 0 & 0 & 0 & 0 & .333 & .25 & 0 & 0 & 250 & 0 & 0 \\
0 & 0 & 0 & .20 & 0 & 0 & 0 & .20 & .20 & 0 & .20 & .333 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .20 & 0 & .20 & .333
\end{pmatrix}$$

Figure (3) Adjacency matrix ***M*** of the graph.

$$\begin{pmatrix}
0.38 & 0.087 & 0.027 & 0 & 0 & .333 & .25 & 0 & 0 & .25 & 0 & 0 \\
0.047 & 0.347 & .21 & 0 & .2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.014 & 0.21 & .347 & .20 & .20 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.027 & 0.087 & .20 & 0 & 0 & 0 & .20 & .20 & 0 & .20 & 0 \\
0.058 & 0.21 & .21 & 0 & .20 & 0 & .25 & .20 & 0 & 0 & 0 & 0 \\
0.142 & 0.017 & 0 & 0 & 0 & .333 & 0 & 0 & 0 & .25 & 0 & 0 \\
0.113 & 0.069 & 0. & 0 & .20 & 0 & .25 & 0 & 0 & .25 & 0 & 0 \\
0. & 0.017 & 0 & .20 & .20 & 0 & 0 & .20 & .20 & 0 & .20 & 0 \\
0 & 0 & 0 & .20 & 0 & 0 & 0 & .20 & .20 & 0 & .20 & .333 \\
0.246 & 0.017 & 0 & 0 & 0 & .333 & .25 & 0 & 0 & 250 & 0 & 0 \\
0 & 0 & 0 & .20 & 0 & 0 & 0 & .20 & .20 & 0 & .20 & .333 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .20 & 0 & .20 & .333
\end{pmatrix}$$

Figure (4) Adjacency matrix ***M*** after two iterations of the algorithm.

$$\left(\begin{array}{cccccccccccc}
.448 & .080 & .023 & 0 & .068 & .426 & .359 & 0 & 0 & .432 & 0 & 0 \\
.018 & .285 & .228 & .007 & .176 & .006 & .033 & .005 & 0 & .007 & 0 & 0 \\
.005 & .223 & .290 & .022 & .173 & 0 & .010 & .017 & .003 & .001 & .003 & .001 \\
0 & .018 & .059 & .222 & .049 & 0 & .002 & .188 & .138 & 0 & .138 & .098 \\
.027 & .312 & .314 & .028 & .450 & .005 & .052 & .022 & .002 & .010 & .002 & .001 \\
.116 & .007 & .001 & 0 & .005 & .158 & .085 & 0 & 0 & .131 & 0 & 0 \\
.096 & .041 & .012 & 0 & .036 & .084 & .197 & .001 & 0 & .104 & 0 & 0 \\
0 & .012 & .042 & .172 & .029 & 0 & .003 & .198 & .133 & 0 & .133 & .096 \\
0 & .001 & .014 & .258 & .009 & 0 & 0 & .267 & .328 & 0 & .328 & .348 \\
.292 & .021 & .002 & 0 & .017 & .323 & .262 & 0 & 0 & .318 & 0 & 0 \\
0 & .001 & .015 & .259 & .009 & 0 & 0 & .268 & .228 & 0 & .328 & .348 \\
0 & 0 & .001 & .037 & .001 & 0 & 0 & .038 & .069 & 0 & .069 & .113
\end{array}\right)$$

Figure (4) Adjacency matrix ***M*** after four iterations of the algorithm.

$$\left(\begin{array}{cccccccccccc}
.822 & .030 & .009 & 0 & .027 & .822 & .822 & 0 & 0 & .822 & 0 & 0 \\
0 & .081 & .200 & 0 & .077 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & .082 & .079 & 0 & .076 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & .027 & 0 & 0 & 0 & .027 & .026 & 0 & .026 & .026 \\
0 & .807 & .807 & 0 & .807 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
.004 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
.002 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & .017 & 0 & 0 & 0 & .018 & 0 & 0 & .018 & .018 \\
0 & 0 & 0 & .482 & 0 & 0 & 0 & .482 & .482 & 0 & .482 & .482 \\
.183 & .002 & 0 & 0 & 0 & .175 & .174 & 0 & 0 & .175 & 0 & 0 \\
0 & 0 & 0 & .482 & 0 & 0 & 0 & .482 & .482 & 0 & .482 & .482 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right)$$

Figure (5) Adjacency matrix ***M*** after 8 iterations of the algorithm.

$$\begin{pmatrix}
1.0 & 0 & 0 & 0 & 0 & 1.0 & 1.0 & 0 & 0 & 1.0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.0 & 1.0 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & .50 & 0 & 0 & 0 & .50 & .50 & 0 & .50 & .50 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & .50 & 0 & 0 & 0 & .50 & .50 & 0 & .50 & .50 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

Figure (6) the output of the enhanced MCL algorithm (the clusters).

4.3-Complexity Analysis of EMCL

The expansion step of ***EMCL*** has complexity $\boldsymbol{O(n^3)}$, assuming some small bound on the expansion exponents ***e***. The inflation has complexity $\boldsymbol{O(n^2)}$. However, the matrices $\boldsymbol{M_i}$ are generally very sparse, or at least the vast majority of the entries are near zero. Pruning in ***EMCL*** involves setting near-zero matrix entries to zero, and can allow sparse matrix operations to improve the speed of the algorithm vastly $\boldsymbol{O(n^3)}$, where ***n*** is the number of vertices. $\boldsymbol{n^3}$ cost of one matrix multiplication on two matrices of dimension ***n***. Inflation can be done in $\boldsymbol{O(n^2)}$ time. The number of steps to converge is not proven, but experimentally shown to be ~10 to 100 steps, and mostly consist of sparse matrices after the first few steps. Speed may be improved through pruning. Inspect matrix and set small values directly to zero (assume they would have reached there eventually anyways). Works well when the diameter of the clusters is small. (Nonhomogeneous distributions of weight).

5-EMCL and RRW

The ***RRW*** is another graph clustering algorithm in which every cluster is stored and the clusters that overlap more than the threshold are later compared, and the lower ranking clusters will be removed, in ***EMCL*** algorithm, the clusters are normally clustered but in case of repeated overlapping, the ***EMCL*** algorithm may fail in finding the clustering of the graphs but this is rarely occurred, so in some situations the ***RRW*** algorithm maybe better than ***EMCL*** algorithm.

6-Conclusion Analysis of EMCL

The ***EMCL*** algorithm scales well with increasing graph size. Works with both weighted and unweighted graphs. Produces good clustering results. Robust against noise in graph data. Number of clusters not specified ahead of time, but can adjust cluster granularity with parameters.Can not find overlapping clusters (in general). Not suitable for clusters with large diameter.

References

[1] Ulrik B., Marco G., Dorothea W. (2003). "Experiments on Graph Clustering", Springer-Verlag Berlin Heidelberg, pp. 568-579.

[2] Andrew D. K., , (2004) "Graph Clustering with Restricted Neighborhood Search", MSc. Thesis, Dept. of computer science, University of Toronto.

[3] Pierre L., Etienne B., Christophe A., , (2008) "Bayesian Methods for Graph Clustering", Research Report No. 17.

[4] Olena M., (2007) "Computing Lexical Chains with Graph Clustering", Proceedings of the ACL 2007 Student Research Workshop, pp 85–90.

[5] Schaeffer S. E., (2007). "Graph clustering", C O M P U T E R S C I E N C E R E V I EW 1, Elsevier, pp 27-64 .

[6] G. Gan, C. Ma, J. Wu, , (2007). "Data Clustering: Theory, Algorithms, and Applications", SIAM, Philadelphia**.**

[7] L. R. Rabiner, , (1989) "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, Vol. 77, No. 2, pp. 257-286.

[8] Andreas Noack, (2007). " Energy Models for Graph Clustering", Journal of Graph Algorithms and Applications, vol. 11, no. 2, pp. 453-480 .

[9] Stijn Marinus van Dongen, , (2000). "Graph clustering by FLOW SIMULATION", PhD. Thesis, University of Utrecht, Holland.

[10] Lars Elden, , (2007). "Fundamentals of Matrix Methods in Data Mining and Pattern Recognition", SIAM, Philadelphia.

[11] Ulrike von Luxburg, , (2006). "A Tutorial on Spectral Clustering", Max Planck Institute for Biological Cybernetics.

[12] Chris Biemann, , (2005). "Chinese Whispers: an Efficient Graph Clustering Algorithm and its Application to Natural Language Processing", University of Leipzig, NLP Department, unpublished.

[13] H. S. Shon, S. Kim. Chung, S. Rhee, K. H. Ryu, (2007). "Clustering Approach Using MCL Algorithm for Analyzing Microarray Data",International Journal of Bioelectromagnetism,pp 65-66

[14] Bose N. K., Liang P., , (1996). "Neural Network Fundamental with Graphs, Algorithms and Applications", McGraw-Hill.

خوارزمية العنقدة MCL المطورة

كاظم مهدي هاشم مؤيد عبد هاني

جامعة ذي قار - كلية التربية – قسم علوم الحاسبات

الخلاصة

إن الهدف من عنقدة بيانات المخططات هو تقسيم العقد في المخططات الكبيرة إلى عناقيد مختلفة أعتمادا على معايير مختلفة كدرجة أرتباط العقدة والتجاور المتشابه. إن تقنيات عنقدة المخططات مفيدة في أكتشاف المجاميع الكثيفة في المخططات. في هذا البحث، قمنا بتقديم خوارزمية لعنقدة المخططات، هذه الخوارزمية تعتمد خوارزمية ماركوف للعنقدة **Markov Clustering** والتي تستعمل خاصية الأنسياب البياني الأحصائي. قيم التصاعد، المصفوفة وقيمة العتبة جرى إلحاقها بطريقة تضمن ملائمتها لعمل الخوارزمية. بعد ذلك تم تقديم التحليل النظري لإثبات إن خوارزمية ماركوف المطورة **Enhanced *EMCL*-Cluster** تضمن الوصول إلى عنقدة مثالية في حالة وجود تشابه بين العقد يضمن ذلك. كذلك تمت مقارنة خوارزمية العنقدة المقترحة مع خوارزميات العنقدة اللأخرى.