International Journal of Mathematical Archive-6(10), 2015, 76-81
Available online through www.ijma.info ISSN 2229 – 5046

# CLUSTERING CATERPILLAR GRAPH: AN ALGORITHM

**ISHWAR BAIDARI*[1], AJITH HANAGAWADIMATH[2], H. S. RAMANE[3]**

**[1]Associate Professor, Department of Computer Science, Karnatak University, Dharwad, India.**

**[2]Research scholar, Department of Computer Science, Karnatak University, Dharwad, India.**

**[3]Professor, Department of Mathemaics, Karnatak University, Dharwad, India.**

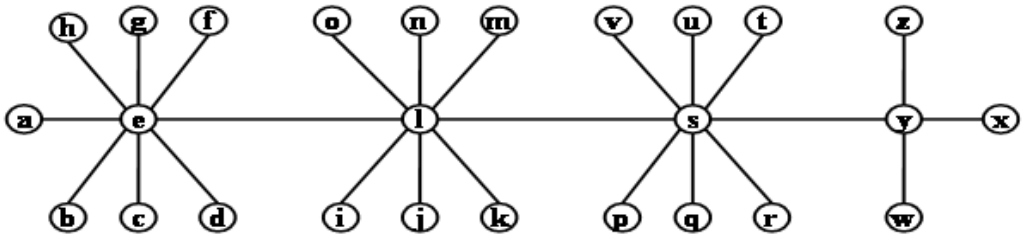
***(Received On: 19-09-15; Revised & Accepted On: 19-10-15)***

***ABSTRACT***

***A*** *Caterpillar graph C(Pk) is a tree having a chord less path Pk, called the backbone that contains at least one end point of every edge. Edges connecting the leaves with the backbone are called hairs. In a complete caterpillar graph each vertex of its backbone has a nonempty set of hairs denoted by CC(Pk) a complete caterpillar graph with backbone Pk. In this paper we propose a simple and efficient algorithm for clustering caterpillar graph vertex as base.*

***Key words:*** *Caterpillar graph, Degree, Vertex, Queue, Cluster, Color.*

## INTRODUCTION




Clustering can be considered the most important unsupervised problem so as every other problem of this kind. It deals with finding a structure in a collection of unlabeled data. A cluster is therefore a collection of objects which are similar between them and are dissimilar to the objects belonging to other clusters. The process of identifying this structure in terms of grouping the data elements is called clustering, also called data classification [1].

Graphs are formed by a set of vertices and set of edges denoted by G= (V, E). Edges connect between pairs of vertices. Graph clustering is the task of grouping the vertices of the graph into clusters taking into consideration the edge structure of the graph in such a way that there should be many edges within each cluster and relatively few between the clusters. Graph clustering in the sense of grouping the vertices of a given input graph into clusters, here we consider a specific class of graph called Caterpillar graph.

Some literature [2, 3, 4, 5, 6, 7] addressed clustering the sets of graphs based on structured similarity, but in caterpillar graph we designed an algorithm which find clusters within a caterpillar graph.

**Some of the properties of Caterpillar graph [8]:**

**Lemma 1:** If Pk is a chord less path with k vertices then, $m(Pk)= m(Pk-2) - m(Pk-3)$, $k \geq 4$ with $m(P_1) = 1$, $m(P_2) = 2$ and $m(P_3) = 2$. Two vertices are twins in a graph if they have the same neighborhood. Jou et al proved the following properties.

***Corresponding Author: Ishwar Baidari*[1]***
***[1]Associate Professor, Department of Computer Science, Karnatak University, Dharwad, India.***

**Lemma 2:** If x and y are twins in a graph G then m(G) = m(G-x), m(G) = m(G-y).

**Lemma 3:** If it is an induced sub graph of G, then m(H) ≤ m(G).

**Lemma 4:** For any two disjoint graphs U and Z in (U∪Z) = m(U) . m(Z).

Let V(Pk) = {V1, V2,…….Vk}. For each Vi∈V(Pk), H(Vi) is the set of its pendent vertices and |H(Vi)| is an independent set but it is not maximal in C(Pk). If some vertex of H(Vi) belongs to a MCS then every vertex of H(Vi) must belong to it otherwise it is not maximal of two vertices of H(Vi) are twins in C(Pk). We can construct them into a single vertex that represent a whole set H(Vi), i=1,2,….,k. Let Gk be the construction group of C(Pk) otherwise that is also a caterpillar graph with at most one pendent vertex at each Vi.

## ALGORITHM FOR CLUSTERING A CATERPILLAR GRAPH

Here we propose a simple and efficient algorithm for clustering caterpillar graph.

**Concept used to form Clusters*:***

In the following algorithms, clustering is done based on the degree of vertices present in the graph. Degree of a vertex 'u' is nothing but the number of edges originating from + terminating into the vertex 'u'.First the degree of all the vertices is found and the vertices with degree > = 2 are placed into a First-in First-out Queue. The vertices placed into the queue will be the cluster heads and the clusters will be formed around these vertices.

Once all the Cluster Head vertices are determined and placed into the queue, the formation of clusters begins around each of those Cluster Head vertices. The vertices that will become the members of the clusters will be the vertices whose degree = 1. In this way the clustering of a given caterpillar graph will be done.

**Some Results on clustering of caterpillar graph:**

*The degree of a vertex u is the number of edges incident to it and is denoted by deg(u).*

**Theorem 1:** Let *CPk* be the caterpillar with *n* vertices. Let $u_1, u_2, \ldots, u_k$ be the cluster heads of *CPk*, $k \le n$. Then $\sum_{i=1}^{k} \deg(u_i) = n + k - 2$.

**Proof:** Let $u_1, u_2, \ldots, u_n$ be the vertices of the caterpillar. Let $u_1, u_2, \ldots, u_k$, be the cluster heads of the caterpillar such that the vertex $u_i$ is adjacent to $u_{i+1}$, $i = 1, 2, \ldots, k-1$ and $k \le n$.

Therefore the vertex $u_1$ has $\deg(u_1) - 1$ leaves, $u_k$ has $\deg(u_k) - 1$ leaves and $u_i$ has $\deg(u_i) - 2$ leaves, $i = 2, 3, \ldots, k-1$.

We know that for any tree with *n* vertices

$$\sum_{i=1}^{n} \deg(u_i) = 2(n-1)$$

Therefore

$$\sum_{i=1}^{k} \deg(u_i) + \sum_{i=k+1}^{n} \deg(u_i) = 2(n-1)$$

$$\sum_{i=1}^{k} \deg(u_i) + (\deg(u_1) - 1) + \sum_{i=2}^{k-1} (\deg(u_i) - 2) + (\deg(u_k) - 1) = 2(n-1)$$

$$2\sum_{i=1}^{k} \deg(u_i) - 2 - 2(k-2) = 2(n-1)$$

$$\sum_{i=1}^{k} \deg(u_i) = n + k - 2.$$

**Example:**

For the caterpillar above, $n = 10$ and $k = 4$.

$$\sum_{i=1}^{4} \deg(u_i) = n + k - 2 = 10 + 4 - 2 = 12 .$$

The complete bipartite graph $K_{1,t}$ is called a star in which one vertex has degree equal to $t$ and all other vertices has degree one.

**Theorem 2:** Every cluster of a caterpillar is a star $K_{1,t}$, ($t \geq 0$).

**Proof:** For clustering in a caterpillar, the vertex having degree greater than or equal to two is identified as a cluster head and the vertices having degree one which are adjacent to cluster head are the pendant vertices. Without loss of generality, let $u_1$ be the one of the cluster head and $u_2$, $u_3$, …,$u_k$ be the pendant vertices adjacent to $u_1$. Then the induced subgraph induced by the vertices $u_1$, $u_2$, …,$u_k$ form a star $K_{1,t}$. Thus every cluster of caterpillar is a star.

*The graph G – e is obtained from G by removing its edge e.*

**Theorem 3:** Let $e = (u, v)$ be an edge of the backbone of the caterpillar *CPk* where $\deg(u) \geq 2$ and $\deg(v) \geq 2$. Then *CPk – e* is a disconnected graph with two components, each is again a caterpillar.

**Proof:** Let $e = (u, v)$ be an edge of the backbone of the caterpillar *CPk*, where $\deg(u) \geq 2$ and $\deg(v) \geq 2$. Also $e = (u, v)$ lies on the unique path. Removing *e* from *CPk*, disconnects the graph into two components. It is obvious to see that each of these components is again a caterpillar.

**ALGORITHM**

Cluster (G)

1. foreach vertex u ∈ G.V
2. u.color = Green
3. u.degree = 0
4. SetLabel (u, Not-Finished )
5. foreach edge (u,v) ∈ G.E
6. (u,v).color = Green
7. SetLabel( (u,v), Undetermined )
8. Head_Queue=ϕ
9. Head_count= 0
10. foreach vertex u ∈ G.V
11. if (GetLabel (u)= = Not-Finished)
12. Degree = 0
13. foreach v ∈G.Adj [u]
14. Degree = Degree + 1
15. u.degree = Degree
16. Set Label( u , Finished )
17. if(u.degree> = 2 )
18. u.color = Red
19. Enqueue( Head_Queue , u )
20. Head_Count=Head_Count+ 1
21. Cluster_Number = 0
22. while(Head_Queue≠ϕ)
23. u = Dequeue( Head_Queue )
24. Cluster_Number = (Cluster_Number + 1)
25. Create set :Set_Cluster_Number {ϕ}
26. Add ‘u’ to set Set_Cluster_Number
27. foreach v ∈Adj[ u ]
28. ifv.degree = = 1
29. Add _v‘ to set Set_Cluster_Number
30. SetLabel( (u, v), Cluster_Edge)
31. else
32. (u, v).color = red
33. SetLabel((u, v), Cluster_Connecting_Edge)

**TIME COMPLEXITY**

1. The initializations of vertices in the for loop of lines 1-4 takes O(V)time.
2. The initializations of edges in the for loop of lines 5-7 takes O(E) time.
3. The for loop of lines 10-20 processes every vertex v∈V to find their respective degrees. For this the adjacency-lists of all the vertices will be scanned once. So the sum of the lengths of all the adjacency lists is as follows:
   $\sum_{v=1}^{V} \left| Adj(v) \right| = 2 * E$
   Hence the time taken for scanning the adjacency lists is O(E).
4. The while loop of lines 22-33 processes all the vertices and edgesonce, and hence it takes O(V+E) time. *Hence*, Adding all the times we get :
   O(V) + O(E) + O(E) + O(V+E) = O(V+E).

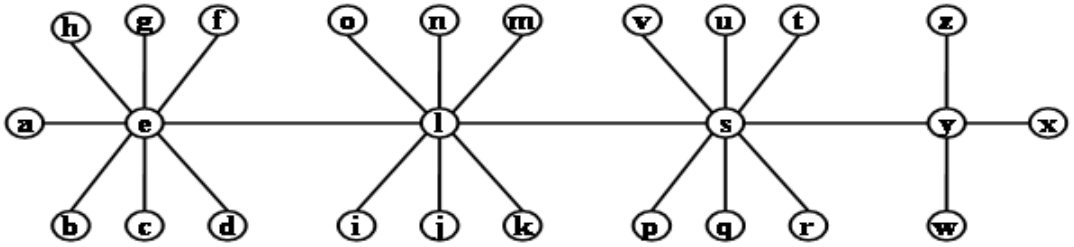
**WORKING OF ALGORITHM**

The working of the above algorithm is as explained below:

Cluster (G):

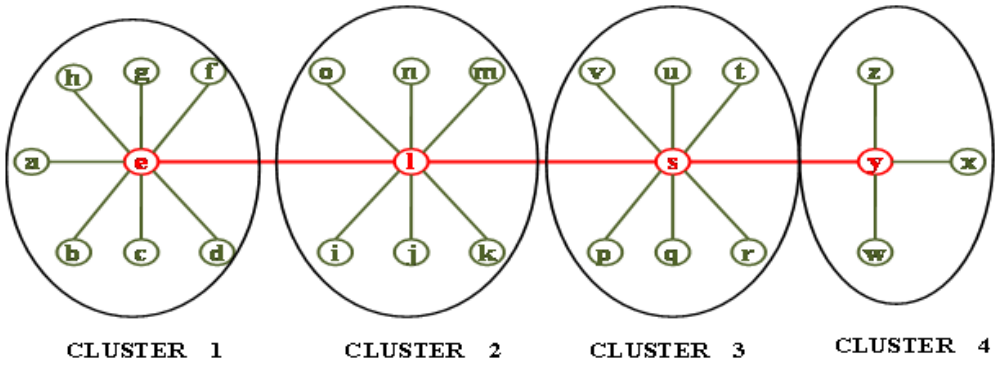
- The for loop in lines 1-4 initializes all the vertices of the given graph. The vertices are colored white in line 2, their degree is set to 0 in line 3 & a label ―Not Finished is set in line 4.
- The for loop in lines 5-7 initializes all the edges of the given graph. Their color is set to Green in line 6 & a label is set as ―Undetermined in line 7.
- In line 8, a First-in First-out Queue by name“ Head Queue” is declared and initialized with ϕ. This queue is used to hold all the Cluster Head vertices.
- In line 9, a variable Head Count is declared and initialized with 0. This is used to count the total number of clusters formed.
- The for loop in lines 10-20 finds the degree of every vertex v ∈V. The condition in line 11 checks whether the label of the vertex is Not-Finished, if this condition is true then its degree will be calculated. In line 12 a variable “Degree” is declared and initialized with 0.The for loop in lines 13-14 finds the degree of vertex ‘u’ by scanning its adjacency list. The value of Degree variable is incremented by 1 in line 15 for each vertex present in the adjacency list.
- The value of the variable Degree is assigned to the u. degree attribute of vetex ‘u’ in line 15 & in line 16 a label is set for vertex ‘u’ as “Finished”.
- The section of the procedure in lines 17-20 determines Cluster Headvertices. The condition in line 17 checks whether the degree of vertex ‘u’ is >=2, if so then its color is changed to red in line 18 & is enqueued into the Head Queuein line 19 and Head Countvariable is incremented by 1 in line 20.
- The while loop in lines 22-33 forms the clusters around each Cluster Head vertices.
- The condition in line 22 ensures that the while loop repeats until the Head Queuebecomes empty.
- First element of Head Queue is dequeued and assigned to ‘u’ in line 23. Line 25 creates a set data-Structure according to the value of the variable Cluster Number in line 24, and the vertex ‘u’ is added to the set created in line 26.
- The for loop in lines 27-33 determines all Cluster member vertices & also distinguishes the edges as either “Cluster Edge” or “Cluster Connecting Edge”. The condition in line 27ensures that the for loop repeats until there is a vertex remaining in the u‘s adjacency list. The condition in line 28 checks whether the degree of vertex ‘v’ = 1, if so then vertex ‘v’ is added to the set created in line 29 & the edge (u, v) is labeled as “Cluster Edge” in line 30. Otherwise the color of edge (u, v) is changed to red in line 32 & a label is set to it as“ Cluster Connecting Edge” in line 33.

**Example 1:** Consider the Caterpillar graph given below:

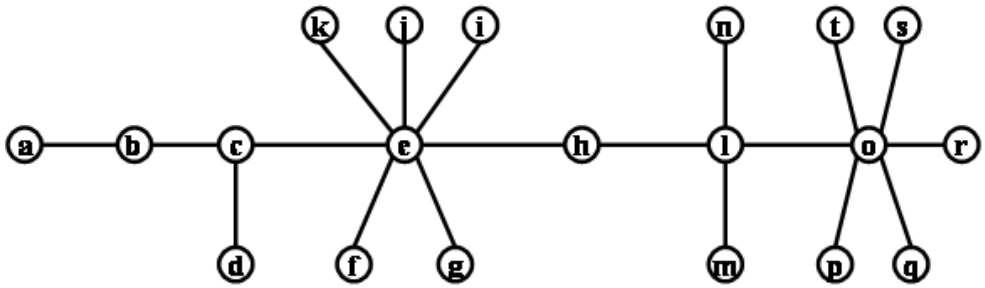


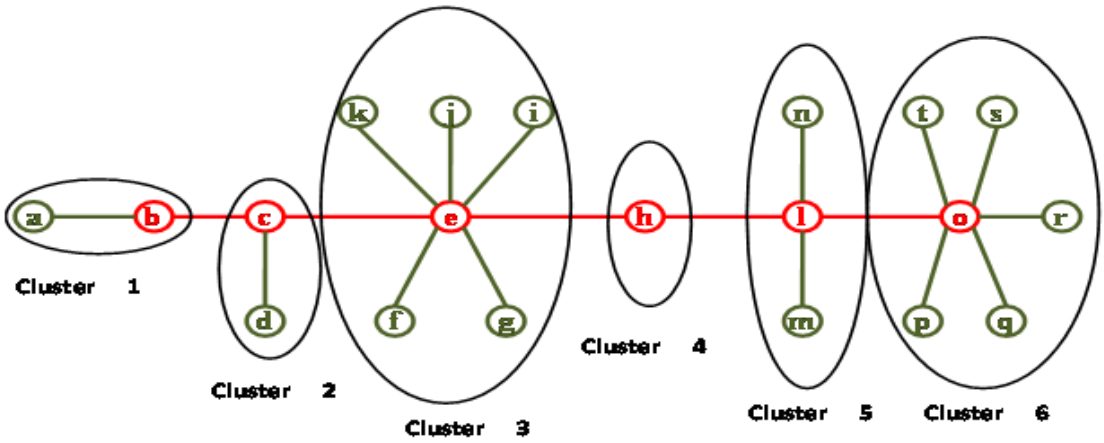
After running the above algorithm the output will be as follows:




**Example 2:** Consider the caterpillar graph given below:



After running the above algorithm the output will be as follows:




## REFERENCES

1. J.M.Kleinberg, E Tardos, Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields, journal of the ACM49 (5) (2002) k-23.
2. H. Bunke, P. Foggia, C. Guidobaldi, M. Vento, Graph clustering using the weighted minimum common supergraph, in: E.R.Hancock, M.Vento(Eds.), Proceedings of the Fourth IARP International Workshop on Graph Based Representations in Pattern Recognition, in: Lecture Notes in Computer Science, vol. 2726, Springer Verlag GmbH, Berlin, Heidelberg, Germany, 2003.
3. A. Hlaoui, S. Wang, Median graph computation for graph clustering, Soft Computing—A Fusion of Foundations Methodologies and Applications 10 (1) (2006) 47.
4. B. Luo, R.C. Wilson, E.R. Hancock, Spectral feature vectorsfor graph clustering, in: T. Caelli, A. Amin, R.P. Duin, M. Kamel, D. de Ridder (Eds.), Proceedings of the Joint IARP International Workshops on Syntactical and Structural Pattern Recognition and Statistical Pattern Recognition, in: Lecture Notes in Computer Science, vol. 2396, Springer Verlag GmbH, Berlin, Heidelberg, Germany, 2002.
5. B. Luo, R.C. Wilson, E.R. Hancock, Spectral clustering of graphs, in: G. Goos, J. Hartmanis, J. van Leeuwen (Eds.), Proceedings of the Tenth International Conference on Computer Analysis of Images and Patterns, CAIP, in: Lecture Notes in Computer Science, vol. 2756, Springer Verlag GmbH, Berlin, Heidelberg, Germany, 2003.

6. A. Robles Kelly, E.R. Hancock, Graph edit distance from spectralseriation, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (3) (2005) 365–378.
7. S.E. Schaeffer, Algorithms for nonuniform networks, Ph.D. Thesis, Helsinki University of Technology, Espoo, Finland, April 2006.
8. Carmen Ortiz, Monica Vilanueva, Maximal independent sets in caterpillar graphs Discrete Applied Mathematics, 160(2012) 259-266.

**Source of support: Nil, Conflict of interest: None Declared**