**Algorithm dividing graph into clusters by using graph cut.**

**Habiba El-Zohny(1) ,Safaa Radwan(2) Hend El- Morsy (3)**

1. , (2) Mathematics Department

Faculty of science

Alazhar University

Cairo - Egypt

**Abstract: In** this paper we will introduce an algorithm which divide graph into clusters by using graph cut. The desirable cluster will be computed before and after cut and compare the two states. Another algorithm for min-cut will be illustrated for weighted and un weighted graph. The adjancy matrix for min-cut will be discussed.

**Definitions:**

**Cluster graph:** Is a group of nodes in a graph that are more connected to each other than they are to the rest of the graph [1].

**Graph cut:** A cut is a partition of node set V into two parts S,S'= V – S and is defined by the set of edges ( I , j ) such that i , the cost of cut ( S, S' )is given as CS,S' = . The st-cut is cut satisfying the properties s

**Density of graph G(V,E) :** Is the ratio of the number of edges present to the maximum possible, *d*(G) =  *,* for , we set d(G) = 0 . A graph of density one is called *complete* [3].

**Desirable cluster:** Is a cluster which its internal density is greater than the density of graph[4].

**Main Results:**

For graph G(V,E) with n vertices if we want to divide this graph into clusters we can use min-cut as illustrated in the following algorithm,

First we select an edge from edges of lowest density then we cut this edge to separate the two nodes , continue cutting edges with lowest density until we obtain separated clusters from given graph .

**Note:** the input graph is connected while the output is disconnected component.

We will introduce an algorithm describes these iterations .

**The Algorithm:**

**Input**: connected graph G with n vertices.

**Output**: disconnected cluster component C1 , C2 , C3 ,……C n.

**Steps:**

1. Select an edge { n1, n2} from area of lowest density between two clusters.
2. Cut this edge to separate n1 , n2 .
3. If such edge exist then ,

Go to step 1

Else.

1. Output C1 ,C2 …….Cn.

End while

End algorithm.

**Example 1:** For graph shown in Fig.(1) by applying the above algorithm we have:

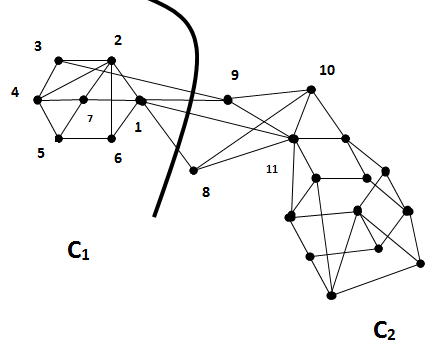
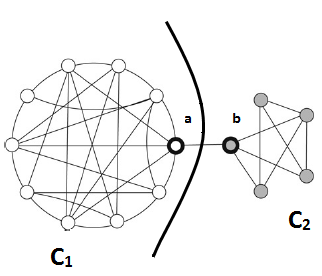


Fig.(1)

Steps:

1. Select edge {1 , 11}.
2. Cut this edge then ,
3. Go to step 1 with edges { 1,9} , {3,9} , {1,8}.
4. Cut selected edges in each step.
5. Output C1 , C2.

**Example 2:** For simple example of graph shown in Fig.(2) we have:



**Steps:** Fig.(2)

1. Select the edge { a,b }.
2. Cut this edge .
3. Output C1,C2.

**Graph cut and desirable clusters:**

In previous chapter[4] we computed desirable cluster for cluster graph ,

we can apply the previous algorithm to cut clusters into separated components by considered that each component is separated graph.

**Example 3:**For graph shown in Fig.(3) , compute desirable cluster befor and after using graph cut algorithm.

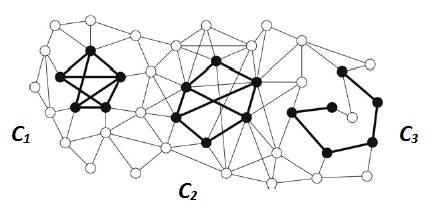


Fig.(3)

1. **Before applying the algorithm :**

First we will compute the density of graph *d(G):*

*d(G) =*  = 0.073

Then compute the density of C1 , C2 , C3 :

*d(C1) = =* 0.4 , then the ratio *Ȓ1* = = 0.1825

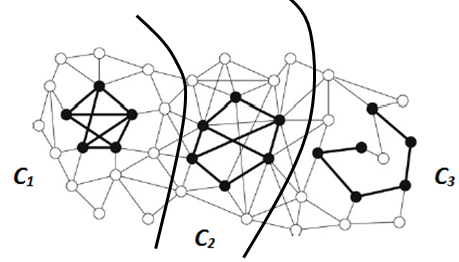
*d(C2)* = = 0.3 , ratio *Ȓ2* = = 0.1666

*d(C3) = =*0.2 , ratio *Ȓ3* = = 0.438

Since *Ȓ1*>  *Ȓ2*>  *Ȓ3* then C1 is the best cluster.

1. **After using graph cut:**

As shown in Fig.(4) we consider each component as separated graph we have:



Density of G1 d(G1) = 25/(14 . 13) = 0.13736 .

Density of cluster C1 d(C1) = 8/(5 .4) = 0.4 .

*Ȓ1*= 0.13736/0.4 = 0.3434 .

d(G2)= 32/(15 .14 ) = 0.15238 .

d(C2)= 8/(6 .5) = 0.3 . *Ȓ2 =* 0.50793 .

d(G3) = 18/(15 . 14) = 0.08571 .

d(C3)= 5/ (6 .5 ) = 0.2 . *Ȓ3* = 0.42855 .

Since *Ȓ1*>  *Ȓ3*>  *Ȓ2* then C1 is also the best cluster.

**Example 4:** For graph shown in Fig(4) compute desirable cluster before and after cut.

****

Fig.(4)

1. **Before applying the algorithm :**

d(G)= 40/ ( 21 . 20 ) = 0.09523 .

d(C1)= 11/ ( 7 . 6) = 0.2619 . *Ȓ1*= 0.3636 .

d(G2)= 24/ (13 . 12 ) = 0.1538 .

d(C2)= 22/ (12 . 11) = 0.1666 . . *Ȓ2*= 0.9231 .

Since *Ȓ1*>  *Ȓ2* then C1 is the best cluster.

1. **After using graph cut:**

d(G1)= 14/42 = 0.333 .

d(C1)= 13 / 30 = 0.433 . *Ȓ1*= 0.769 .

d(G2) = 24/ 90 = 0.266 .

d(C2) = 22/72 = 0.3055 . *Ȓ2*= 0.872 .

Since *Ȓ1*>  *Ȓ2* then C1 is also the best cluster.

From the previous examples we find that desirable cluster doesn't change before and after cuts.

**Theorem 1:**

For cluster graph G(V , E) , min-cut doesn't change the desirable cluster.

**Proof :**

The proof comes directly from the above discussion.

**Random min-cut algorithm and its adjancy matrix:**

There are two algorithms to find min-cut for graph , first for un weighted graph similar to Monte – Carlo min-cut algorithm [5] , and the other for weighted graph.

1. **Un weighted graph:**

**Input:** Connected un weighted graph G ,|v| ≥ 2.

**Algorithm body:**

1. Select edge e randomly from G and contract it.
2. Remove self loop from G.
3. If |v| = 2 then,

Output |E|

Else

1. Go to step 1.

End while

Output |E|

Where |E| is min – cut edge.

**Example 5:** Find min – cut for graph shown in Fig(5).

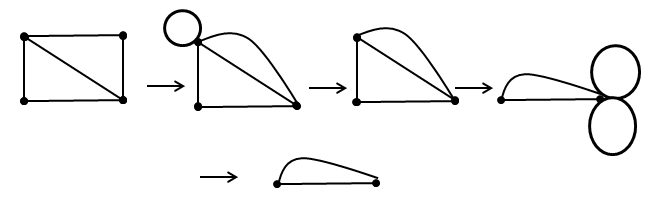


Fig.(5)

Note that we apply 2 iterations to get min – cut.

**Lemma 1:**

For connected (weighted or un weighted ) graph G with n vertices, there are (n-2) iterations to get min – cut of this graph.

1. **Weighted graph:**

**Input:** Connected un weighted graph G ,|v| ≥ 2.

Steps:

1. Select edge e with min – weight We randomly from outer edges.
2. Remove self loop (of maximum weight).
3. Merge edge e and add We to We' , for e' is neighbor edge of e.
4. If |v| = 2 then,

Output |E|

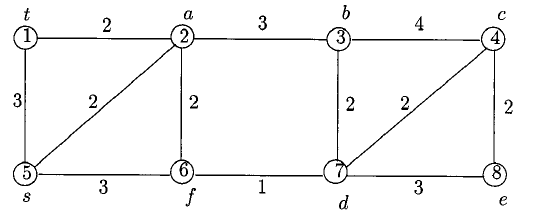
Else

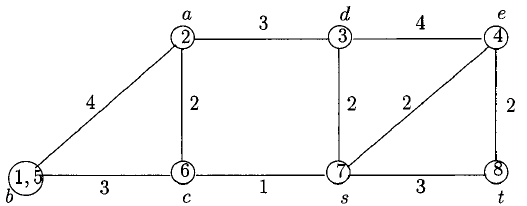
1. Go to step 1.

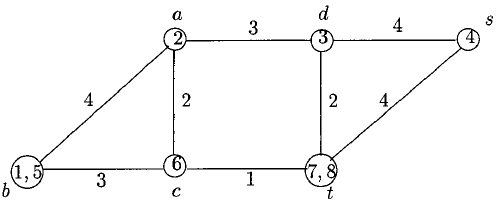
End while

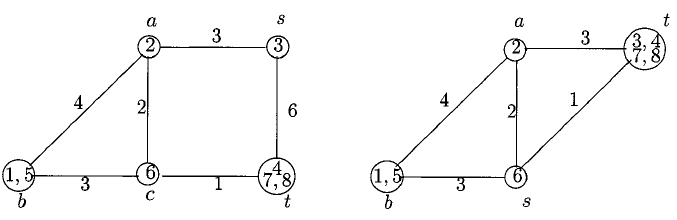
Output |E|

**Example 6:**For graph shown in Fig.(6) apply the previous algorithm.

****



****

****

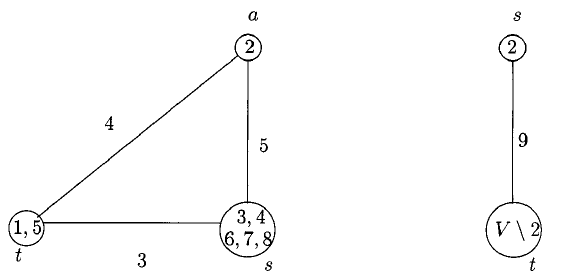
****

Fig.(6)

From the previous example this graph of 8 vertices we have 6 iterations (8-2) to get min – cut.

**Example 7:** For graph of 5 vertices shown in Fig.(7) we have 3 iterations to get min – cut.

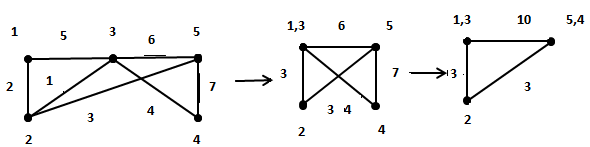
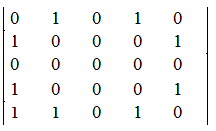
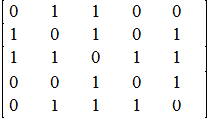
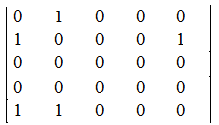
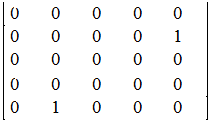
****

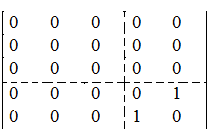
Fig.(7)

The adjancy matrix for min – cut for this graph is shown in Fig.(8)

** **

After replacement the modification of min – cut:

****

From the previous example we find that all final elements of adjancy matrix after cut will be zeros except the elements of min –

**References**

[1] Wilson R.J.:Introduction to graph theory. Oliver and Boyed, Edinburgh 1972.

[2] Pushmeet , Kohli , Philip H.S.Torr, : Efficiently solving dynamic Markov random fields using graph cuts , EPSRC research grant (May 2005).

[3] Satu Elisa ,: Graph clustering , Computer Science Review, (2007) ,27-64.

[4] Elzohny.H, Elmorsy.H ,:Algorithm for computing desirable cluster , Alazhar university, Cairo ,2013.

[5] paper B.