An Algorithm for Partitioning Community Graph into Sub-community Graphs Using Graph Mining Techniques

Bapuji Rao and Anirban Mitra

Abstract Using graph mining techniques, knowledge extraction is possible from the community graph. In our work, we started with the discussion on related definitions of graph partition both mathematical as well as computational aspects. The derived knowledge can be extracted from a particular sub-graph by way of partitioning a large community graph into smaller sub-community graphs. Thus, the knowledge extraction from the sub-community graph becomes easier and faster. The partition is aiming at the edges among the community members of different communities. We have initiated our work by studying techniques followed by different researchers, thus proposing a new and simple algorithm for partitioning the community graph in a social network using graph techniques. An example verifies about the strength and easiness of the proposed algorithm.

Keywords Adjacency matrix · Cluster · Community · Graph partition · Sub-Graph

1 Introduction

We use graph theory's some important techniques to solve the problem of partitioning a community graph to minimize the number of edges or links that connect different community [1]. The aim of partitioning a community graph to sub-graphs is to detect similar vertices which form a graph and such sub-graphs can be formed. For example, considering Facebook is a very large social graph. It can be

B. Rao (✉) · A. Mitra
Department of CSE and IT, V.I.T.A.M., Berhampur, Odisha, India
e-mail: rao.bapuji@gmail.com

A. Mitra
e-mail: mitra.anirban@gmail.com

partitioned into sub-graphs, and each sub-group should belong to a particular characteristics. Such cases we require graph partitions. In this partition, it is not mandatory that each sub-group contain similar number of members. A partition of a community graph is to divide into clusters, such that each similar vertex belongs to one cluster. Here a cluster means a particular community. Based on this technique, we partition a community graph into various sub-graphs after detecting various vertices belonging to a particular community or cluster.

2 Basics in Graph Theory

Social network, its actors and the relationship between them can be represented using vertices and edges [2]. The most important parameter of a network (i.e., a digraph) is the number of vertices and arcs. Here we denote n for number of vertices and m for number of arcs. When an arc is created by using two vertices u and v, which is denoted by uv. Then the initial vertex is the u and the terminal vertex is the v in the arc uv.

2.1 Digraph

A digraph or directed graph $G = (V, A)$ with $V = \{V_1, V_2, \ldots\ldots.., V_n\}$ can be represented as adjacency matrix A. The matrix A is of order nXn where A_{ij} is 1 or 0 depending on V_iV_j is an edge or not. Note that $A_{ii} = 0$ for all i.

2.2 Sub-digraph

A sub-digraph of G to be (V_1, A_1) where $V_1 \subseteq V, A_1 \subseteq A$ and if uv is an element of A_1 then u and v belong to V_1.

2.3 Adjacency Matrix

Let a graph G with n nodes or vertices $V_1, V_2, \ldots.., V_n$ having one row and one column for each node or vertex. Then the adjacency matrix A_{ij} of graph G is an nXn square matrix, which shows one (1) in A_{ij} if there is an edge from V_i to V_j; otherwise zero (0).

2.4 Good Partition

When a graph is divided into two sets of nodes by removing the edges that connect nodes in different sets should be minimized. While cutting the graph into two sets of nodes so that both the sets contain approximately equal number of nodes or vertices [1].

In Fig. 1 graph G_1 has seven nodes $\{\boldsymbol{V}_1, \boldsymbol{V}_2, \boldsymbol{V}_3, \boldsymbol{V}_4, \boldsymbol{V}_5, \boldsymbol{V}_6, \boldsymbol{V}_7\}$. After cutting into two parts approximately equal in size, the first partition has nodes $\{\boldsymbol{V}_1, \boldsymbol{V}_2, \boldsymbol{V}_3, \boldsymbol{V}_4\}$ and the second partition has nodes $\{\boldsymbol{V}_5, \boldsymbol{V}_6, \boldsymbol{V}_7\}$. The cut consists of only the edge $(\boldsymbol{V}_3, \boldsymbol{V}_5)$ and the size of edge is 1.

In Fig. 2 graph G_2 has eight nodes $\{\boldsymbol{V}_1, \boldsymbol{V}_2, \boldsymbol{V}_3, \boldsymbol{V}_4, \boldsymbol{V}_5, \boldsymbol{V}_6, \boldsymbol{V}_7, \boldsymbol{V}_8\}$. Here two edges, $(\boldsymbol{V}_3, \boldsymbol{V}_7)$ and $(\boldsymbol{V}_2, \boldsymbol{V}_6)$ are used to cut the graph into two parts of equal size rather than cutting at the edge $(\boldsymbol{V}_5, \boldsymbol{V}_8)$. The partition at the edge $(\boldsymbol{V}_5, \boldsymbol{V}_8)$ is too small. So we reject the cut and choose the best one for cut consisting of edges $(\boldsymbol{V}_2, \boldsymbol{V}_6)$ and $(\boldsymbol{V}_3, \boldsymbol{V}_7)$, which partitions the graph into two equal sets of nodes $\{\boldsymbol{V}_1, \boldsymbol{V}_2, \boldsymbol{V}_3, \boldsymbol{V}_4\}$ and $\{\boldsymbol{V}_5, \boldsymbol{V}_6, \boldsymbol{V}_7, \boldsymbol{V}_8\}$.

2.5 Normalized Cuts

A good cut always balance the size of cut itself against the sizes of the sets of created cut [1]. For this normalized cut method is being used. First it has to define the volume of set of nodes or vertices V which is denoted as Vol (V) is the number of edges with at least one end in the set of nodes or vertices V.

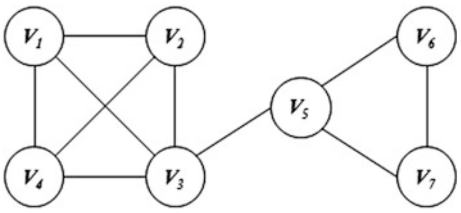


Fig. 1 Graph G_1 with seven nodes

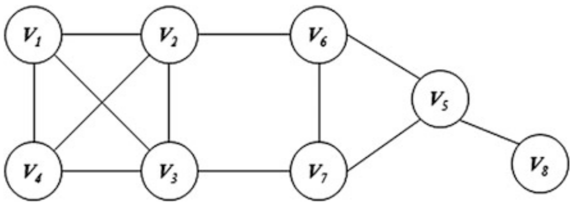


Fig. 2 Graph G_2 with eight nodes

Let us partition the nodes of a graph into two disjoint sets say A and B. So the Cut (A, B) is the number of edges from the disjoint set A to connect a node in the disjoint set B. The formula for normalized cut values for disjoint sets A and B = Cut (A, B)/Vol (A) + Cut (A, B)/Vol (B).

2.6 Graph Partitions

Partition of graph means a division in clusters, such that similar kinds of vertices belong to a particular cluster [1]. In a real world vertices may share among different communities. When a graph is divided into overlapping communities then it is called a cover.

A graph with K-clusters and N-vertices, the possible number of Stirling number of the second kind is denoted as S(N, K). So the total number of possible partitions is said to be the Nth Bell number is given with the formula $B_N = \sum_{K=0}^{N} \mathrm{S}(N, K)$ [3]. When the value of N is large then B_n becomes asymptotic [4].

While partitioning a graph having different levels of structure at different scales [5, 6], the partitions can be ordered hierarchically. So in this situation cluster plays an important role. Each cluster displays the community structure independently, which consists of set of smaller communities.

Partitioning of graph means dividing the vertices in a group of predefined size. So that the frequently used vertices are often combined together to form a cluster by using some techniques. Many algorithms perform a partition of graph by means of bisecting the graph. Iterative bisection method is employed to partition a graph into more than two clusters and this algorithm is called as Kernighan-Lin [7]. The Kernighan-Lin algorithm was extended to extract partitions of graph in any number of clusters [8].

Another popular bisection method is the spectral bisection method [9, 10], is completely based on the properties of spectrum of the Laplacian matrix. This algorithm is considered as quiet fast. According to Ford and Fulkerson [11] theorem that the minimum cut between any two vertices U and V of a graph G, is any minimum number of subset of edges whose deletion would separate U from V, and carries maximum flow from U to V across the graph G. The algorithms of Goldberg and Tarjan [12] and Flake et al. [13, 14] are used to compute maximum flows in graphs during cut operation. Some other popular methods for graph partition are level-structure partition, the geometric algorithm, and multilevel algorithms [15].

3 Proposed Algorithms and Analysis

```
Algorithm  Community_Graph_Partition()
// Global Declarations
n    : Number of Communities.
NCM  [1:n,1:2]:  Holds  community  number  and  number  of
community members of each community.
tcm      : To count total number of community members.
CMM  [1:tcm+1,  1:tcm+1]:  Adjacency  matrix  of  Community
Members of order tcmXtcm.

i.[Read Community Data]
   Call Read_Community_Data().
ii.[Generate and assign every members code]
    CallAssign_Community_Member_Codes()
iii.[Creation of adjacency matrix of all the members]
     Call Community_Member_Matrix()
iv. [Partition of Community Graph]
     Call Graph_Partition()
v. (a) Set s:=0.
   (b) Repeat For I:=1, 2,.........,n:
        (1) s:= s+NCM[I][1].
     [Show the 'I'th sub-community graph after partition]
        (2) Call Sub_Community_Matrix_Display(s).
       End For
vi. Exit.

Procedure-I.Read_Community_Data()
i. Set tcm:=0.
ii. Read Number of communities as 'n'.
iii.Read  Community  details  such  as  community  code  and
number  of  members  of  each  community,  and  assign  to  the
matrix NCM[][].
iv. Repeat For I:=1, 2, ......., n:
      tcm := tcm + NCM[I][2].
    End For
v. Return.

Procedure-II.Assign_Community_Member_Codes()
i.   Set K := 1.
ii.  Set Pro := 1.
iii. Repeat For I :=1, 2,......., n:
       (1) If NCM[I][1]>=1 AND NCM[I][1]<=9,
           Then (a) Pro := 10.
           Else If NCM[I][1]>=10 AND NCM[I][1]<=99,
           Then (b)Pro:= 100.
           Else If NCM[I][1]>=100 AND NCM[I][1]<=999,
           Then(c)Pro:=1000.
           Else
               (d)Break.
           End If
       (2)Repeat For J := 1, 2,......, NCM[I][2]:
           (a) Set CMM[1][K+1] := (NCM[I][1]*10) + J.
           (b) Set CMM[K+1][1] := (NCM[I][1]*10) + J.
           (c) K := K + 1.
          End For
     End For
iv. Return.
```

Procedure-III.Community_Member_Matrix()

```
i. Get the edge data of all the community members.
ii. Store the above data in the matrix CMM[ ][ ].
iii. Return.
```

Procedure-IV.Graph_Partition()

```
i. Repeat For I := 1, 2, ........, tcm+1:
ii. Repeat For J := 1, 2, ........, tcm+1:
       If CMM[1][J+1]/Pro ≠ CMM[I+1][1]/Pro,
       Then
           [Cut off edge between communities of Different
            group of communities]
              Set CMM[I+1][J+1] := 0.
       End If
     End For
   End For
iii. Return.
```

Procedure-V.Sub_Community_Matrix_Display(size)

```
size: Size of each community.
i.   Set  x:=0.
ii.  Set count := x.
iii. Repeat For i:=count, count+1,....,size:
iv.     Repeat For j:=count, count+1,....,size:
        (a)If i=count And j=count, Then: Display "C".
           Else If i=count Andj≠count,
           Then: Display CMM[1][j].
           Else If I≠countAnd j=count,
           Then: Display CMM[i][1].
           Else: Display CMM[i][j].
           End If
        End For
        (b)x:=x+1.
     End For
v. x:=x-1.
vi. Return.
```

3.1 Explanation

The proposed algorithm consists of five procedures. Procedure-I allows to read the details about number of communities and number of community members of all the communities. In this example the output has been derived after implemented using C++ programming language. The data related to community and their edges are read from two data files namely “*commun1.txt*” and “*graph.dat*”. Procedure-II which generates and assigns community member codes. Procedure-III creates the community adjacency matrix. Procedure-IV allows us to partition the community adjacency matrix by assigning ‘0’ over ‘1’ which indicates the edge between the community members of dissimilar communities. Finally Procedure-V displays every community’s adjacency matrix. From the adjacency matrices we can draw the community sub-graphs.

3.2 Example

We propose a community graph [16, 17] with 22 individual communities from four different communities $\{C_1, C_2, C_3, C_4\}$ which is shown in Fig. 3. We try to partition

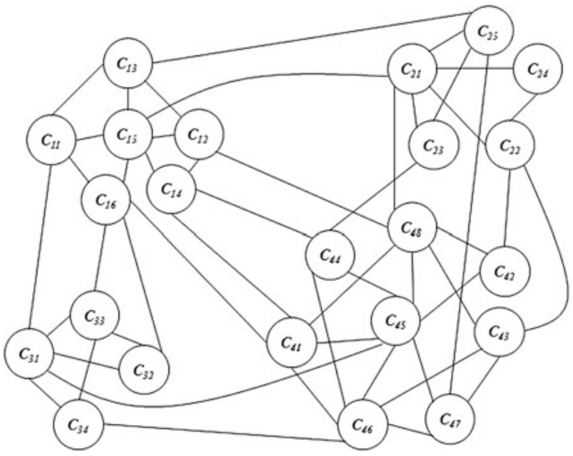
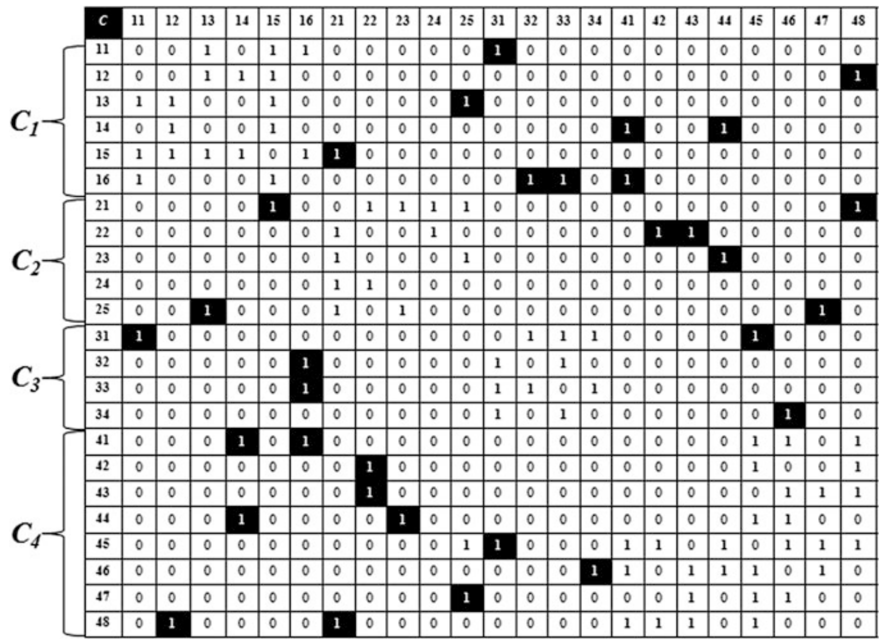
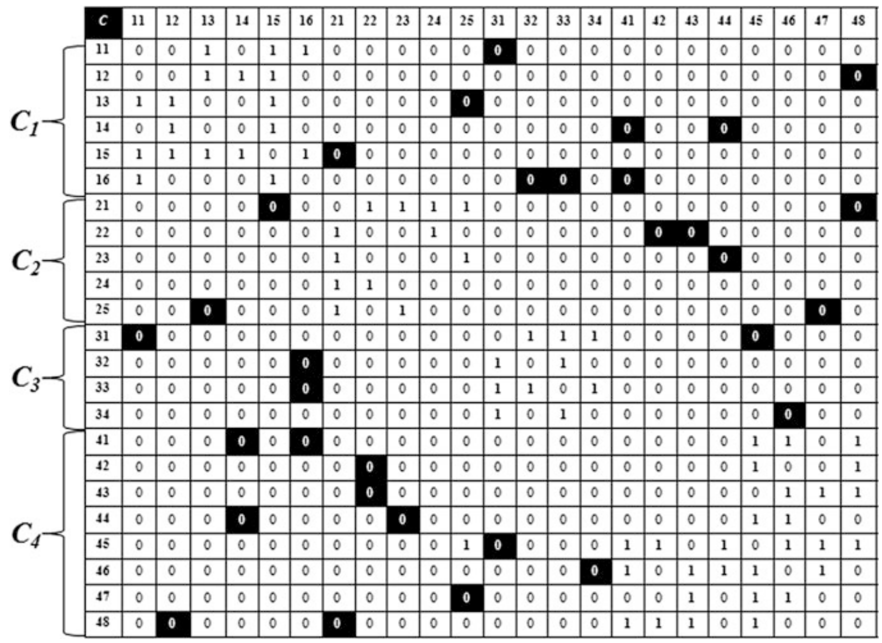


Fig. 3 Community graph of communities $\{C_{11}\ldots C_{16},\ C_{21}\ldots C_{25},\ C_{31},\ldots C_{34}, C_{41}\ldots C_{48}\}$



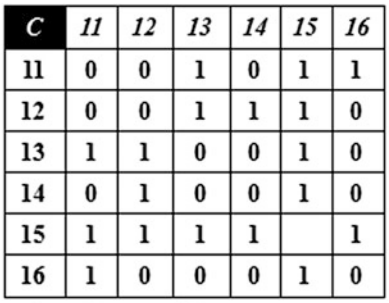
	C	11	12	13	14	15	16	21	22	23	24	25	31	32	33	34	41	42	43	44	45	46	47	48
C_1	11	0	0	1	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	12	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	13	1	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	14	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
	15	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	1	0	0	0	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0
C_2	21	0	0	0	0	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1
	22	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0
	23	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
	24	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	25	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0
C_3	31	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	0	0	0
	32	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
	33	0	0	0	0	0	1	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0
	34	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0
C_4	41	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
	42	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
	43	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	44	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0
	45	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	0	1	0	1	1	1
	46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	1	0
	47	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	1	0	0
	48	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0

Fig. 4 Adjacency matrix of community graph in Fig. 3



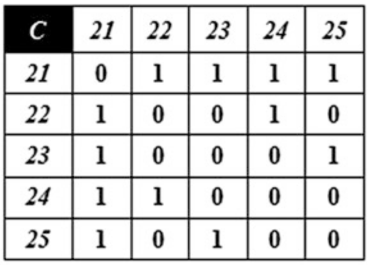
	C	11	12	13	14	15	16	21	22	23	24	25	31	32	33	34	41	42	43	44	45	46	47	48
C_1	11	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	12	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	13	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	14	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C_2	21	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
	22	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	23	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	24	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	25	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C_3	31	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
	32	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
	33	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0
	34	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
C_4	41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
	42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
	43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	44	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
	45	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0	1	1	1
	46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	0
	47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0
	48	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0

Fig. 5 Adjacency matrix of community graph after cut-off edges between community members of dissimilar communities



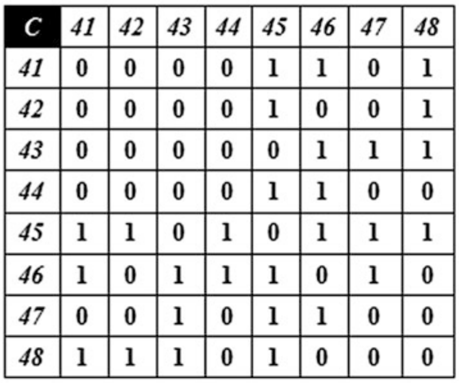
C	11	12	13	14	15	16
11	0	0	1	0	1	1
12	0	0	1	1	1	0
13	1	1	0	0	1	0
14	0	1	0	0	1	0
15	1	1	1	1		1
16	1	0	0	0	1	0

i) C_1 Adjacency Matrix



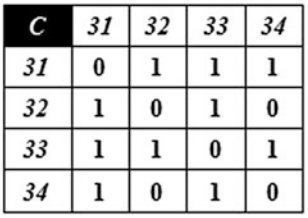
C	21	22	23	24	25
21	0	1	1	1	1
22	1	0	0	1	0
23	1	0	0	0	1
24	1	1	0	0	0
25	1	0	1	0	0

ii) C_2 Adjacency Matrix



C	41	42	43	44	45	46	47	48
41	0	0	0	0	1	1	0	1
42	0	0	0	0	1	0	0	1
43	0	0	0	0	0	1	1	1
44	0	0	0	0	1	1	0	0
45	1	1	0	1	0	1	1	1
46	1	0	1	1	1	0	1	0
47	0	0	1	0	1	1	0	0
48	1	1	1	0	1	0	0	0

iv) C_4 Adjacency Matrix



C	31	32	33	34
31	0	1	1	1
32	1	0	1	0
33	1	1	0	1
34	1	0	1	0

iii) C_3 Adjacency Matrix

Fig. 6 Adjacency matrices of communities C_1, C_2, C_3, and C_4

this graph into four sub-graphs of communities $\{C_1, C_2, C_3, C_4\}$. We try to represent this graph in memory in an adjacency matrix form by following graph techniques which is shown in Fig. 4. Then we try to locate edges between communities members formed from two different communities.

The black filled boxes indicate the edge between the community members of dissimilar communities which is indicated in Fig. 5. These edges are considered as edges between dissimilar communities. So these edges must be cut. Once such edges are cut, then the original graph can be partitioned into so many sub-graphs. And we can say that the graph has been partitioned across edges of community

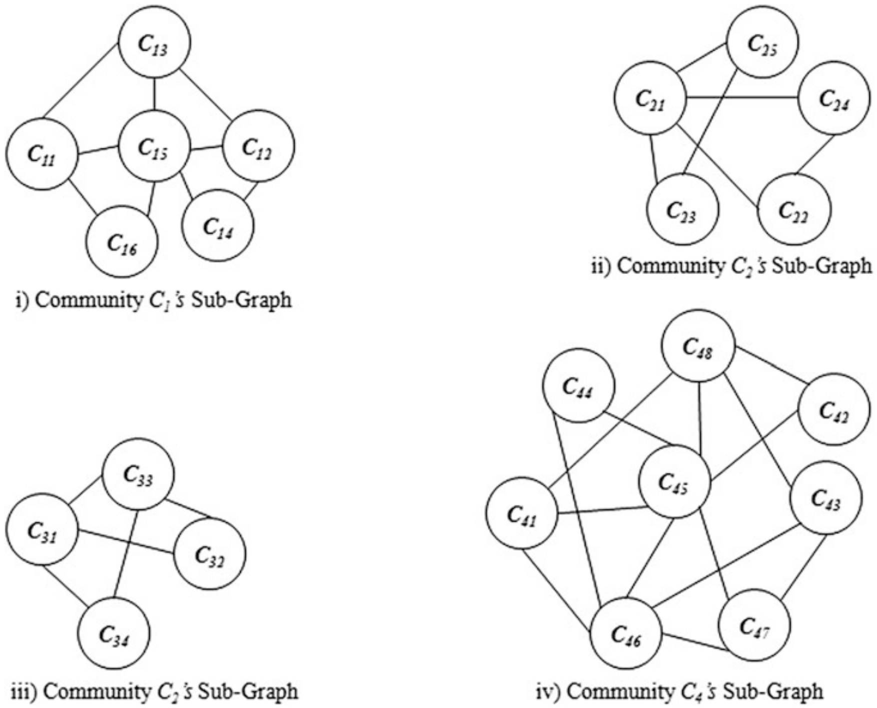
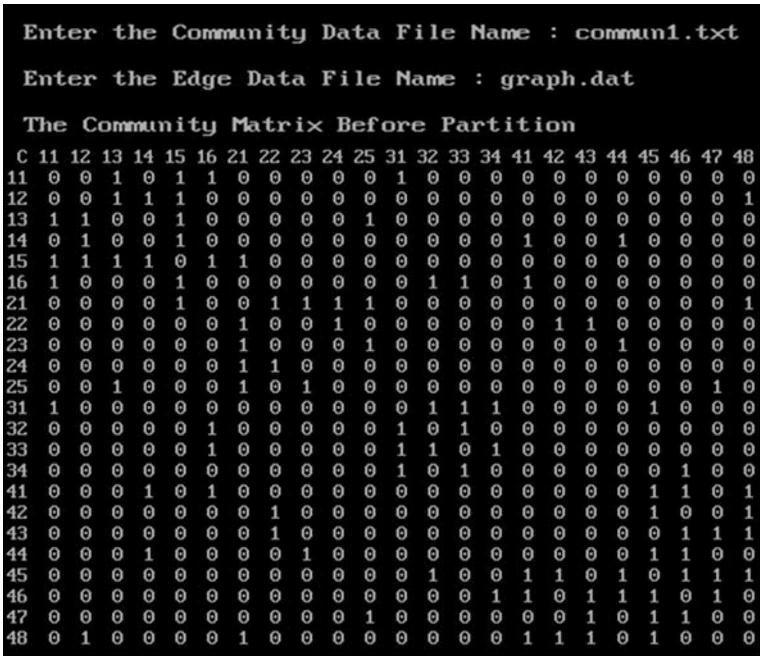

C13
C11
C15
C12
C16
C14
i) Community C1's Sub-Graph
C25
C21
C24
C23
C22
ii) Community C2's Sub-Graph
C33
C31
C32
C34
iii) Community C2's Sub-Graph
C48
C44
C42
C45
C41
C43
C46
C47
iv) Community C4's Sub-Graph

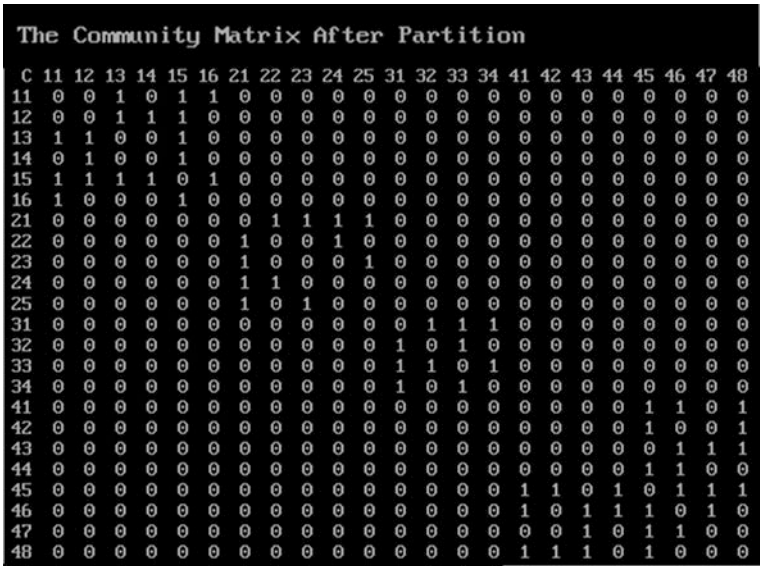
Fig. 7 Communities $\{C_1, C_2, C_3, C_4\}$'s sub-graphs

members of dissimilar communities. To do the edge cut operation, we assign 0 over 1 in the black filled boxes of adjacency matrix in Fig. 5. So that we can say there is no physical edge between those community members across the different communities. From the adjacency matrix of Fig. 5, we can construct four different adjacency matrices for the communities C_1, C_2, C_3, and C_4 which is shown in Fig. 6. For C_1 the community members are $\{11, 12, 13, 14, 15, 16\}$. Similarly for C_2, C_3, and C_4 the community members are $\{21, 22, 23, 24, 25\}$, $\{31, 32, 33, 34\}$, and $\{41, 42, 43, 44, 45, 46, 47, 48\}$ respectively. From these four adjacency matrices, now we can construct the sub-graphs which are shown in Fig. 7.

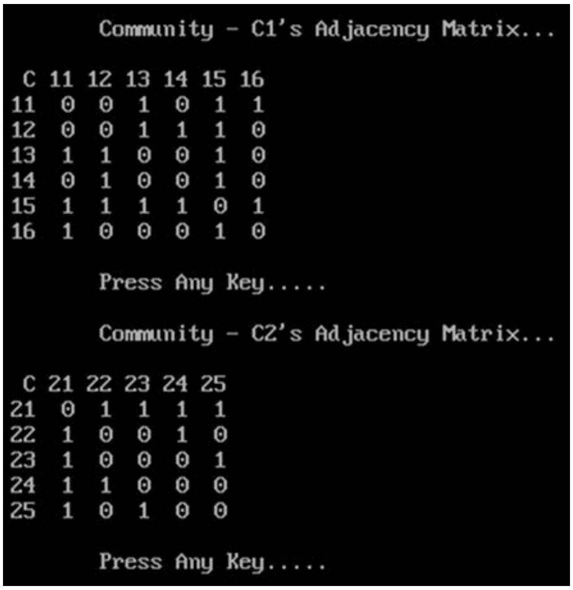
3.3 Output



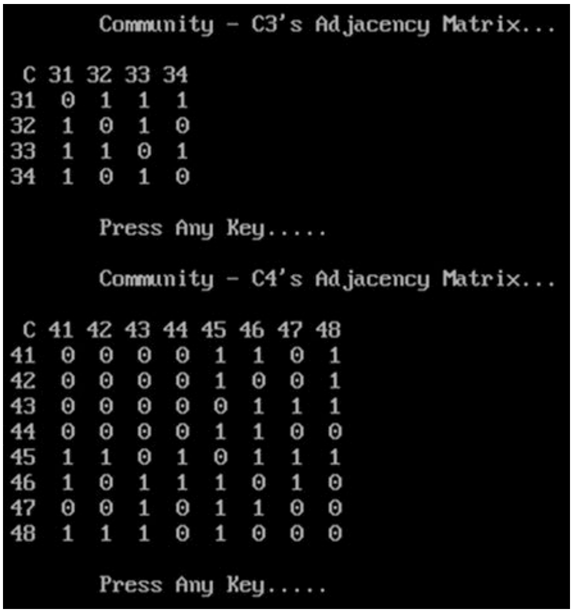
```
Enter the Community Data File Name : commun1.txt

Enter the Edge Data File Name : graph.dat

The Community Matrix Before Partition
 C 11 12 13 14 15 16 21 22 23 24 25 31 32 33 34 41 42 43 44 45 46 47 48
11  0  0  1  0  1  1  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
12  0  0  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
13  1  1  0  0  1  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0
14  0  1  0  0  1  0  0  0  0  0  0  0  0  0  0  1  0  0  1  0  0  0  0
15  1  1  1  1  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
16  1  0  0  0  1  0  0  0  0  0  0  0  1  1  0  1  0  0  0  0  0  0  0
21  0  0  0  0  1  0  0  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  1
22  0  0  0  0  0  0  1  0  0  1  0  0  0  0  0  0  1  1  0  0  0  0  0
23  0  0  0  0  0  0  1  0  0  0  1  0  0  0  0  0  0  0  1  0  0  0  0
24  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
25  0  0  1  0  0  0  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  1  0
31  1  0  0  0  0  0  0  0  0  0  0  0  1  1  1  0  0  0  0  1  0  0  0
32  0  0  0  0  0  1  0  0  0  0  0  1  0  1  0  0  0  0  0  0  0  0  0
33  0  0  0  0  0  1  0  0  0  0  0  1  1  0  1  0  0  0  0  0  0  0  0
34  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0  0  0  0  0  0  1  0  0
41  0  0  0  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  1
42  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1
43  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  1  1  1
44  0  0  0  1  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  1  1  0  0
45  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1  1  0  1  0  1  1  1
46  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  1  1  1  0  1  0
47  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  1  0  1  1  0  0
48  0  1  0  0  0  0  1  0  0  0  0  0  0  0  0  1  1  1  0  1  0  0  0
```



```
The Community Matrix After Partition
 C 11 12 13 14 15 16 21 22 23 24 25 31 32 33 34 41 42 43 44 45 46 47 48
11  0  0  1  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
12  0  0  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
13  1  1  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
14  0  1  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
15  1  1  1  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
16  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
21  0  0  0  0  0  0  0  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0
22  0  0  0  0  0  0  1  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
23  0  0  0  0  0  0  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0
24  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
25  0  0  0  0  0  0  1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
31  0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  0  0  0  0  0  0  0  0
32  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0  0  0  0  0  0  0  0  0
33  0  0  0  0  0  0  0  0  0  0  0  1  1  0  1  0  0  0  0  0  0  0  0
34  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0  0  0  0  0  0  0  0  0
41  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  1
42  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1
43  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  1
44  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0
45  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  0  1  0  1  1  1
46  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  1  1  0  1  0
47  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  1  0  0
48  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  0  1  0  0  0
```



```
        Community - C1's Adjacency Matrix...

 C 11 12 13 14 15 16
11  0  0  1  0  1  1
12  0  0  1  1  1  0
13  1  1  0  0  1  0
14  0  1  0  0  1  0
15  1  1  1  1  0  1
16  1  0  0  0  1  0

        Press Any Key.....

        Community - C2's Adjacency Matrix...

 C 21 22 23 24 25
21  0  1  1  1  1
22  1  0  0  1  0
23  1  0  0  0  1
24  1  1  0  0  0
25  1  0  1  0  0

        Press Any Key.....
```



```
        Community - C3's Adjacency Matrix...

 C 31 32 33 34
31  0  1  1  1
32  1  0  1  0
33  1  1  0  1
34  1  0  1  0

        Press Any Key.....

        Community - C4's Adjacency Matrix...

 C 41 42 43 44 45 46 47 48
41  0  0  0  0  1  1  0  1
42  0  0  0  0  1  0  0  1
43  0  0  0  0  0  1  1  1
44  0  0  0  0  1  1  0  0
45  1  1  0  1  0  1  1  1
46  1  0  1  1  1  0  1  0
47  0  0  1  0  1  1  0  0
48  1  1  1  0  1  0  0  0

        Press Any Key.....
```

4 Conclusions

We have partitioned our large community graph into sub-community graphs using the concepts of graph technique, especially by detecting an edge between the nodes of different communities. Initial portion of the work is a brief review of the literature on graph partition related to mathematical formulae as well as graph mining techniques. A simple graph technique for partition of a large community graph has been proposed. An appropriate example from social community network background has been represented using the graph theoretic concepts. The paper concludes with focusing on process of partitioning a community graph. There after the various sub-community graphs are to be shown in its adjacency matrix format. Hence extracting knowledge from a particular sub-community graph becomes easier and faster.

References

1. Rajaraman, A., Leskovec, J., Ullman, J.D.: Mining of Massive Datasets. Copyright © 2010, 2011, 2012, 2013, 2014
2. Mitra, A., Satpathy, S.R., Paul, S.: Clustering analysis in social network using covering based rough set. In: 2013 IEEE 3rd International Advance Computing Conference (IACC), India, 22 Feb 2013, pp. 476–481, 2013
3. Andrews, G.E.: The Theory of Partitions. Addison-Wesley, Boston, USA (1976)
4. Lovasz, L.: Combinatorial Problems and Exercises. North-Holland, Amsterdam, The etherlands (1993)
5. Ravasz, E., Barabasi, A.L.: Phys. Rev. E **67**(2), 026112 (2003)
6. Ravasz, E., Somera, A.L., Mongru, D.A., Oltvai, Z.N., Barabasi, A.L.: Science **297**(5586), 1551 (2002)
7. Kernighan, B.W., Lin, S.: Bell Syst. Tech. J. **49**, 291 (1970)
8. Suaris, P.R., Kedem, G.: IEEE Trans. Circuits Syst. **35**, 294 (1988)
9. Barnes, E.R.: SIAM J. Alg. Discr. Meth. **3**, 541 (1982)
10. Scholtz, R.A.: The spread spectrum concept. In: Abramson, N. (ed) Multiple Access, Piscataway, NJ: IEEE Press, ch. 3, pp. 121–123 (1993)
11. Ford, L.R., Fulkerson, D.R.: Canadian J. Math. **8**, 399 (1956)
12. Goldberg, A.V., Tarjan, R.E.: J. ACM **35**, 921 (1988)
13. Flake, G.W., Lawrence, S., Giles, C.L.: In: Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (ACM Press, Boston, USA), pp. 150–160 (2000)
14. Flake, G.W., Lawrence, S., Lee Giles, C., Coetzee, F.M.: IEEE Comput. **35**, 66 (2002)
15. Pothen, A.: Graph Partitioning Algorithms with Applications to Scientific Computing. Technical Report, Norfolk, VA, USA (1997)
16. Rao, B., Mitra, A.: A new approach for detection of common communities in a social network using graph mining techniques. In: 2014 International Conference on High Performance Computing and Applications (ICHPCA), pp. 1–6, 22–24 Dec 2014. doi: 10.1109/ICHPCA.2014.7045335
17. Rao, B., Mitra, A.: An approach to merging of two community sub-graphs to form a community graph using graph mining techniques. In: 2014 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC-2014), 978-1-4799-3972-5/14/$31.00 @2014, pp. 460–466, Coimbatore, India, Dec 2014