Journal of Engineering and Applied Sciences 13 (Special Issue 5): 4753-4760, 2018
ISSN: 1816-949X

An Approach to Detect Patterns in a Social Attributed Graph Using Graph Mining Techniques

Bapuji Rao, Sarojananda Mishra and T. Kartik Kumar
Department of Computer Science Engineering and Applications,
Indira Gandhi Institute of Technology (IGIT), Sarang, Odisha, India

Abstract: In a social attributed graph nodes have attributes labeled with a type of work title. So, every node is considered as object having its own work title as its attribute. The researchers detect user query pattern as sub-graphs in the social attributed graph using graph mining techniques. In this scenario the researchers propose university graph as a social attributed graph which compromises of various nodes having its attributes labeled with a type of work title such as Dean, Associate Dean, Professor and Associate Professor. Using the above listed attributes and the actual number of nodes, the authors design the proposed model as undirected graph. From this attributed graph, the authors detect line and loop pattern (or query). For this the authors propose an algorithm to detect line and loop pattern from the university attributed graph using graph mining techniques.

Key words: Attribute, attributed graph, sub-graph, pattern, loop pattern, mining techniques

INTRODUCTION

In real life graphs, attributes associated with vertices represent vertex properties. In social graphs, vertex attributes are used to model personal characteristics. Similarly, in web graph, vertex attributes can be assigned with content such as keywords and tags. This type of extended graph representation is called as attributed graph which may be used for detection of graph patterns that provide relevant knowledge in various applications. From a social attributed graph how graph patterns are detected? This study is about to detect line and loop patterns as sub-graphs from a social attributed graph using graph mining techniques. For this the authors propose university graph as a social attributed graph for detection of line and loop patterns. To detect line and loop patterns as sub-graphs, the authors propose an algorithm called PattGra using graph mining techniques.

Literature review: Graph X-Ray (G-Ray), a fast method that detects sub-graphs which match the exact query pattern or non exact query pattern proposed by Tong *et al.* (2007). This algorithm detects patterns such as line, loop, star and elongated star. To study the correlation between attribute sets and the occurrence of dense sub-graphs in large attributed graphs which is termed as structural correlation pattern mining proposed by Silva *et al.* (2012). The algorithm AGM-DP-TriCycLe proposed by Jorgensen *et al.* (2016) for mining of attributed social graphs with formal privacy guarantees. Georgiy Levchuk *et al.* proposed automated solutions for finding patterns which contains high levels of noise and irrelevant information as well as detection of repetitive patterns and dependencies between entities and attributes in a multi-attributed network data. Algorithm AFGMiner is introduced by Gomes *et al.* (2014) is about to find heavyweight patterns in a dataset of attributed flow graphs. Zhang *et al.* (2014) proposed a method to extract the soft attributed pattern from attributed graph. Rao *et al.* (2015) proposed an algorithm which efficiently detects a common community sub-graph between two community graphs using graph mining techniques. Rao *et al.* (2016) proposed an algorithm to detect a sub-community graph efficiently in n-community graphs using graph mining techniques. In this study the researchers have only focused to detect exact line and loop patterns in the proposed university attributed graph using graph theoretic concepts.

Attributed graph: An attributed graph having n-node X n-Node matrix and an n-Node X m-Attribute Matrix can be defined as:

$$G = \{V = [v_{i,j}],\ A = [a_{i,k}]\}$$

where (i, j) is a pair of nodes with value one, if there exists an edge between them. For every node i which is associated with an attributed vector:

$$a_i = [a_{i,1}, a_{i,2}, a_{i,3}, a_{i,4}, ..., a_{i,n}],\ a_{i,k} = 1$$

Corresponding Author: Bapuji Rao, Department of Computer Science Engineering and Applications,
Indira Gandhi Institute of Technology (IGIT), Sarang, Odisha, India

If node I is labeled with kth attribute value; 0 otherwise can be (Silva *et al*., 2012; Tong *et al*., 2007; Jorgensen *et al*., 2016).

MATERIALS AND METHODS

The researchers propose university attributed graph as social graph (Rao *et al*., 2014; Cook and Holder, 2007) which comprises of nodes with different work titles. These work titles are treated as attributes of nodes in the attributed graph. So, the proposed model has twenty numbers of nodes from 1-20 with four different types of work titles such as Dean, Professor, Associate Professor, and Assistant Professor. The proposed model is depicted in Fig. 1.

The node numbers {1, 2, 8} are assigned with attribute Dean, the node numbers {3, 4, 5, 6, 7, 9} are assigned with attribute Associate Dean, the node numbers {11, 12, 13, 14, 15, 16} are assigned with attribute Professor and the node numbers {10, 17, 18, 19, 20} are assigned with attribute Associate Professor, respectively. Among the nodes there is a relationship which is represented with an undirected edge. Hence, this graph is treated as undirected graph.

According to the definition of attributed graph (Silva *et al*., 2012; Tong *et al*., 2007; Jorgensen *et al*., 2016), the authors have represented the proposed university attributed graph by using two adjacency matrices. The first adjacency matrix is the node-node adjacency matrix which comprises of twenty numbers of nodes and depicted in Fig. 2. Similarly the second adjacency matrix is the node-attribute adjacency Matrix which comprises of twenty numbers of nodes and four numbers of attributes where 1 refers as Dean, 2 refers as Associate Dean, 3 refers as Professor and 4 refers as Associate Professor, respectively and depicted in Fig. 3.

Assume that there are two given patterns such as a line pattern 1-2-3 and a loop pattern 2-3-4-2 depicted in Fig. 4 for detection in the proposed university attributed graph. The numerical code for work titles is depicted in

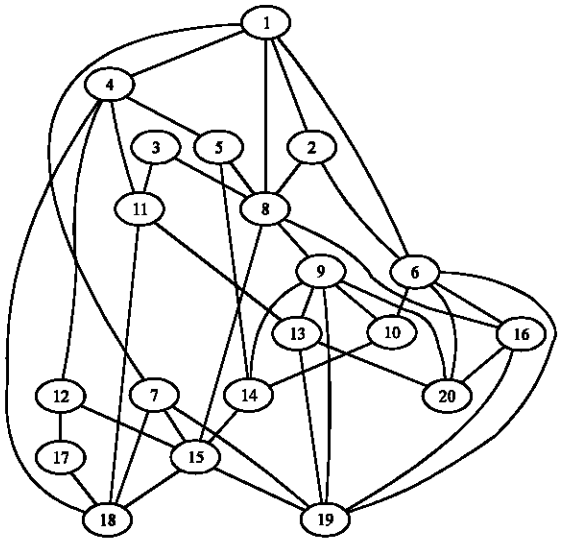

Fig. 1: University attributed graph

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
4	1	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0
5	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
6	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	1
7	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0
8	1	1	1	0	1	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0
9	0	0	0	0	0	0	0	1	0	1	0	0	1	1	0	0	0	0	1	1
10	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0
11	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0
12	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0
13	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1
14	0	0	0	0	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0
15	0	0	0	0	0	0	1	1	0	0	0	1	0	1	0	0	0	1	1	0
16	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1
17	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
18	0	0	0	1	0	0	1	0	0	0	1	0	0	0	1	0	1	0	0	0
19	0	0	0	0	0	1	1	0	1	0	0	0	1	0	1	1	0	0	0	0
20	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	1	0	0	0	0

Fig. 2: Node-node adjacency matrix

	1	2	3	4
1	1	0	0	0
2	1	0	0	0
3	0	1	0	0
4	0	1	0	0
5	0	1	0	0
6	0	1	0	0
7	0	1	0	0
8	1	0	0	0
9	0	1	0	0
10	0	0	0	1
11	0	0	1	0
12	0	0	1	0
13	0	0	1	0
14	0	0	1	0
15	0	0	1	0
16	0	0	1	0
17	0	0	0	1
18	0	0	0	1
19	0	0	0	1
20	0	0	0	1

Fig. 3: Node-attribute adjacency matrix

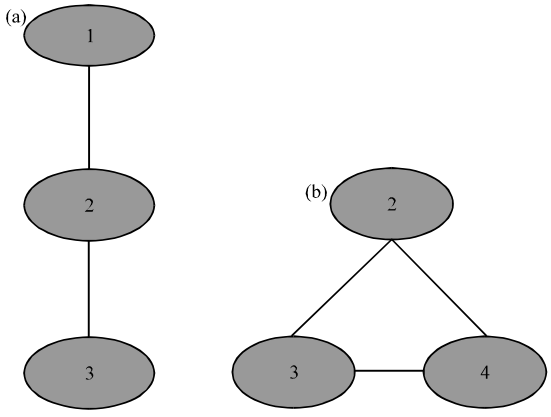

Fig. 4: Patterns to detect: a) Line pattern and b) Loop pattern

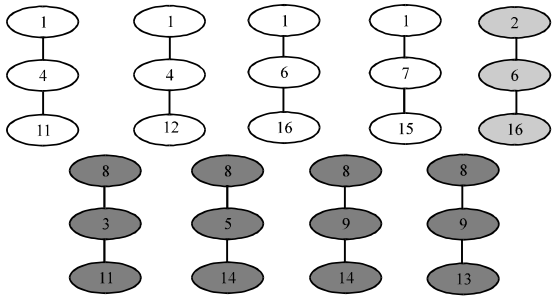

Fig. 5: Nine detected line patterns; Attribut 1-3

Fig. 3, i.e., the 1st row refers such numerical codes 1-4 for Dean, Associate Dean, Professor and Associate Professor, respectively.

To detect line pattern 1-2-3, the researcher store in an array called PAT[]. The array PAT[] start comparing with node-attribute adjacency matrix to count total number of nodes related to all the three attribute codes 1-3. These attributes related nodes and its total number are stored in a matrix MAT[][] which is responsible to generate all the possible node-pairs comprising of all the three attributes related nodes and store in a matrix called Pattern_Matrix[][] with "From node", "To node" and its edge value, i.e., 0 or 1. From this Pattern_Matrix[][], select all the node pairs where the actual edge is present i.e., value 1 and store in a matrix called Result_Marix[][] which only stores "From node" and "To node". Finally, the detected patterns are generated from the Result_Matrix[][] by comparing "To node" and "From node" in a sequence pair of nodes for equality. Such pairs are considered as one line pattern. So, for the given line pattern 1-3, the authors have detected nine line patterns from the university attributed graph and depicted in Fig. 5.

In loop pattern 2-3-4-2, the 1st and last attributes are always same. Hence, the pattern 2-3-4 by ignoring the last attribute i.e., 2. Same process of line pattern is to be followed to get the Result_Matrix[][] which only comprises of "From node" and "To node". Before selecting two pair of nodes as one pattern, the "From Node" of 1st pair of nodes and the "To Node" of 2nd pair of nodes must have an edge i.e., a value 1 in the node-node adjacency matrix. Such pairs are chosen as a loop pattern. So for the given loop pattern 2-3-4-2, the authors have detected eight loop patterns from the university attributed graph and depicted in Fig. 6. For this the authors have proposed an algorithm called PattGra which has capable of detection of both line and loop patterns or sub-graphs from university attributed graph comprising of any three attributes ranging from 1-4.

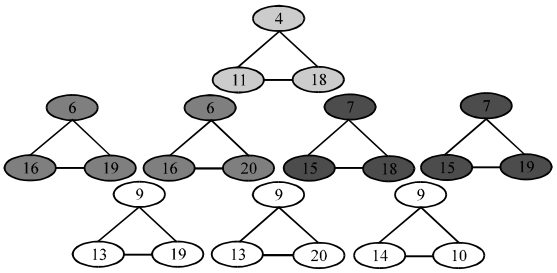

Fig. 6: Eight detected loop patterns; Range 1-4

Algorithm 1; PattGra; Algorithm PattGra (NM, NAM, PAT):

NM[n+1][n+1]: Node-Node Adjacency Matrix where n is the number of nodes
NAM[n+1][m+1]: Node-Attribute Adjacency Matrix, where m is the number of attributes
PAT[4]: Array to hold 3 attributes as pattern for line or 4 attributes as pattern for loop
MAT[3][]: To assign all nodes ID which is related to all three attributes in the given pattern
RMAT[][3]: Matrix to hold node pairs and its edge value
ResultPat[][2]: Matrix to hold the actual node pair to get the result pattern

```
{//check pattern has loop or not
 if(PAT[1]=PAT[4]) then loop:=TRUE; else loop:=FALSE
// get and count all attributes node from NAM[][] and store in MAT[][]
for i:= 1 to 3 do // length of pattern whether loop or no loop
{ ci:=2;
 for j:=1 to m do
 {
  if(PAT[i]=NAM[1][j+1]) then
  {
    for k:=1 to n do
        if(NAM[k+1][j+1]=1) then { MAT[i][ci]:=NAM[k+1][1]; ci:=ci+1
}
   MAT[i][1]:=ci-1; // total number of nodes belong to PAT[i]
  }
 }
}
 // node pair creation from MAT[][] and assign to RMAT[][]
 r:=1;
 for i:=2 to (MAT[1][1]+1) do
 {
  for j:=2 to (MAT[2][1]+1) do
  {
   RMAT[r][1]:=MAT[1][i]; RMAT[r][2]:=MAT[2][j];  r:=r+1
  }
  for j:=2 to (MAT[2][1]+1) do
   for k:=2 to (MAT[3][1]+1) do
    {
     RMAT[r][1]:=MAT[2][j]; RMAT[r][2]:=MAT[3][k];  r:=r+1
    }
 } // outer loop close
 FlagAssignment(NM, RMAT, r)
 PatternDisplay(RMAT, r, MAT, NM)
} // main close
```

Algorithm 2; Procedure for assignment of flag values; Procedure Flag Assignment (NM, RMAT, r):

NM[n+1][n+1]: Node-Node Adjacency Matrix, where n is the number of nodes
RMAT[r][3]: Matrix to hold pair nodes and its flag value

```
{
  for i:=1 to r do
  {
    for j:=1 to n do
    {
     if(RMAT[i][1]=NM[j+1][1]) then break
    }
    for k:=1 to n do
    {
     if(RMAT[i][2]=NM[1][k+1]) then break
    }
    if(NM[j+1][k+1]=1) then RMAT[i][3]:=1; else RMAT[i][3]:=0
  }
}
```

Algorithm 3; Procedure for pattern display; Procedure PatternDisplay(RMAT, r, MAT, NM):

RMAT[r][3]: Matrix to hold pair nodes and its flag value
MAT[3][]: Matrix to hold total number of pattern nodes and its node Ids
NM[n+1][n+1]: Node-Node Adjacency Matrix, where n is the number of nodes
ResultPat[][2]: Matrix to hold the actual node pair where the edge is present
Result[][2]: Matrix to hold the all possible node pairs from it the actual pattern is formed

```
{
  rc:=1;  row:=1;  tnodes:=MAT[1][1]; // total number of base nodes
 // gather all 'from node' to 'to node' with flag value = 1 and assign in ResultPat[][2]
 for i:=1 to r do
 {
   if(RMAT[i][3]=1) then
   {
    ResultPat[row][1]:=RMAT[i][1]
    ResultPat[row][2]:=RMAT[i][2]
    row: = row+1
   }
 }
  for i:=2 to (tnodes+1) do
  {
   for j:=1 to row do
   {
    if(MAT[1][i]=ResultPat[j][1]) then
    {
     Result[rc][1]:=ResultPat[j][1]
     Result[rc][2]:=ResultPat[j][2]
     rc:=rc+1
    }
   }
  }
   pick:=(row-rc)/tnodes; // to find no. of other node pairs other than base number of nodes
   tc:=rc; // total 'to node' comparisons
  // assign 'pick' number of node pairs from bottom of ResultPat[][2] to Result[][2]
  for i:=row-pick to row do
  {
   Result[rc][1]:=ResultPat[i][1]
   Result[rc][2]:=ResultPat[i][2]
   rc:=rc+1
  }
  // to display detected patterns
  for i:=1 to tc do
  {
   for j: = i+1 to rc do
   {
    if(Result[i][2] = Result[j][1]) then
    {
      value: = Verify(Result[i][1], Result[j][2], NM)
     // to display loop pattern
     if(loop=TRUE and value = 1) then
       display(Result[i][1], Result[i][2], Result[j][2], Result[i][1])
     // to display line pattern
```

```
      if(loop=FALSE and value=0) then
        display(Result[i][1], Result[i][2], Result[j][2])
    }
    }
  }
}
```

Algorithm 4; Procedure for verification of loop in the pattern; Procedure verify (row_node, col_node, NM):

```
NM[n+1][n+1]: Node-Node Adjacency Matrix, where n is the number of
nodes
{
  for i: = 1 to n do
    if(row_node = NM[1][i+1]) then break
  for j:=1 to n do
    if(col_node = NM[j+1][1]) then break
  if(NM[i+1][j+1]=1) then return 1; // if loop
  else return 0; // if no loop
}
```

The proposed algorithm PattGra (NM, NAM, PAT) takes three inputs such as the node-node adjacency matrix NM[n][n], the node-attribute adjacency matrix NAM[n][m] and the pattern array PAT (Rao *et al*., 2016) which contains attribute codes from 1-4. If the array PAT (Rao *et al*., 2016) has a loop then the value TRUE; Otherwise, the value FALSE will be assigned to the variable 'loop'.

The proposed algorithm has three procedures namely FlagAssignment( ), PatternDisplay( ) and Verify( ). The procedure FlagAssignment (NM, RMAT, r) passes three arguments such as the matrix NM[n][n], the matrix RMAT[r] (Rao *et al*., 2015) and r which is the actual created number of node-pairs. The matrix NM[][] is the node-node adjacency matrix of the proposed model. The matrix RMAT[][] contains "From node" and "To node" in 1st and 2nd column whereas 3rd column for assignment of flag value i.e. 0 or 1. By comparing every "From node" and "To node" with the matrix NM[][] to find out their actual index positions. Then based on that index positions check out for an edge, i.e., a value 1. If so then assign a value 1 to the 3rd column of matrix RMAT[][]; Otherwise, assign a value 0.

The procedure pattern display (RMAT, r, MAT, NM) passes four arguments. The 1st argument matrix RMAT[][] comprises of "From node", "To node" and flag value. The 2nd argument, r is the actual created number of node-pairs. The 3rd argument matrix MAT[][] comprises of total number of nodes related to the particular attribute of the given pattern and the nodes belong to the particular attribute. The 4th argument matrix NM[][] comprises of n-nodes adjacency edge values i.e., 0s and 1s. From RMAT[][], it starts gathering all the "From node" and "To node" where the flag value is 1 and store in the matrix result Pat[] (Rao *et al*., 2014). Then, compare 1st row of MAT**[1][] with the 1st columns of Result Pat[][1] to get all the nodes of the 1st attribute of the given pattern PAT[4] and store in the matrix result[][2]. Then assign all the common patterns which is made up of the remaining two attributes related nodes can be available from the bottom part of the result Pat[][2] to the matrix result[][2] using the formula pick = (row-rc)/tnodes (refer in procedure pattern display()).

Now, the matrix result[][2] has all the possible detected patterns. Every "To node" from result[][2] of pair-node is being compared with all the "From node" from result[][1] of pair-node. When it is equal then that two nodes result[][1] and result[][2] are passed by calling the procedure verify( ). Then the procedure verify( ) to find the edge value between those nodes. If the edge value between those two nodes is 1 then there is a loop. Otherwise the edge value is 0. Such value is returned by the procedure verify() for every pair-node from the matrix result[][2]. Based on the return value either 1 or 0 by the procedure verify(), the pattern is decided whether it is loop or line.

RESULTS AND DISCUSSION

The researchers have created two datasets for the proposed university attributed graph depicted in Fig. 7. Dataset-1 is the 1st dataset file which comprises of

20	
1	2
1	5
1	6
1	7
1	8
2	1
2	6
2	8
3	8
3	11
4	1
4	5
4	11
4	12
5	4
5	8
5	9

Dataset-1

1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	1
9	2
10	4
11	3
12	3
13	3
14	3
15	3
16	3
17	4
18	4
19	4
20	4

Dataset-2

Fig. 7: Datasets of Fig. 1

Fig. 8: Datasets input

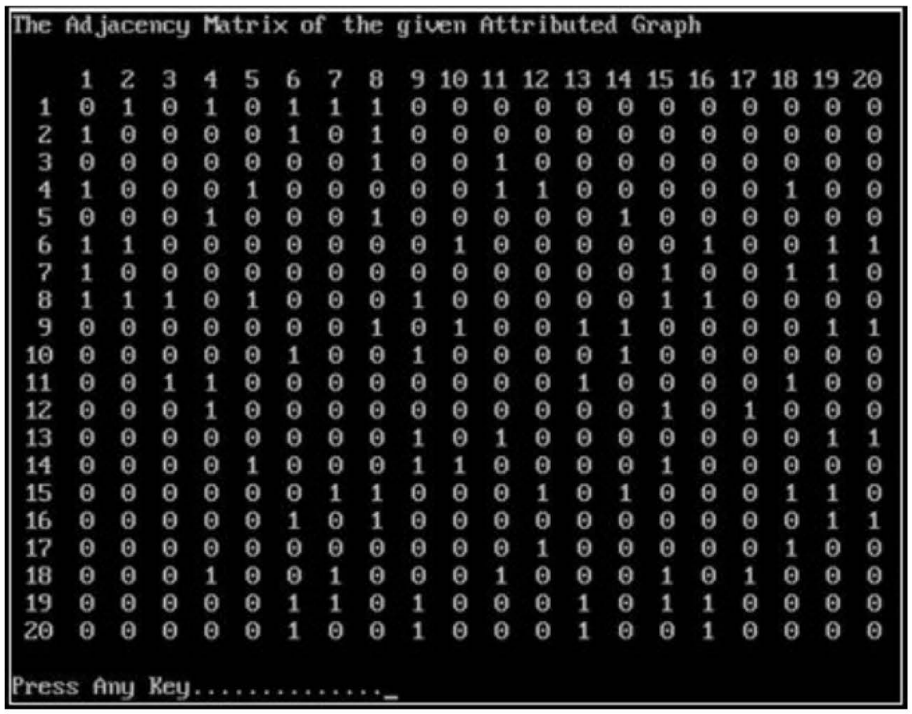

Fig. 9: Node-node adjacency matrix

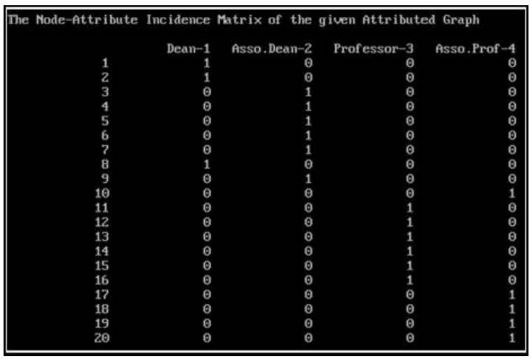

Fig. 10: Node-attribute adjacency matrix

total number of nodes (here it is 20) and "from node" and "to node" is a pair of node where the relation is created (here it is from 2nd row onwards). Dataset-2 is considered as 2nd dataset file which only comprises of node and attributes codes i.e., the node which is related to the attribute. After input these two datasets depicted in Fig. 8, the proposed university attributed graph was successfully represented in the memory using two adjacency matrices namely node-node adjacency matrix and node-attribute adjacency matrix and depicted in Fig. 9 and 10, respectively. The given patterns either line 1-2-3 or loop 2-3-4-2 which is to be detected from the proposed university attributed graph is assigned to the array PAT[4].

After input the line pattern 1-2-3 and the loop pattern 2-3-4-2 to the algorithm, it successfully detected nine line patterns and eight loop patterns which are depicted in Fig. 11 and 12. These detected patterns are graphically depicted in Fig. 5 and 6, respectively. Similarly three more line patterns 1-2-4, 4-3-2 and 3-2-4 were input to the algorithm. It successfully detected twelve patterns of 1-2-4. They were 1-4-18, 1-6-10, 1-6-19, 1-6-20, 1-7-18, 1-7-19, 2-6-10, 2-6-19, 2-6-20, 8-9-10, 8-9-19 and 8-9-20. For pattern 4-3-2, the detected three patterns were 10-14-5, 17-12-4 and 18-11-3. Similarly, for the input pattern 3-2-4, the detected five patterns were 12-4-18, 13-9-10, 14-9-19, 14-9-20 and 16-6-10.

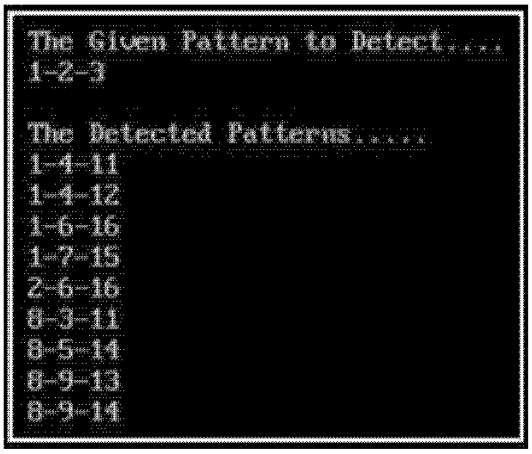

Fig. 11: Nine detected line patterns

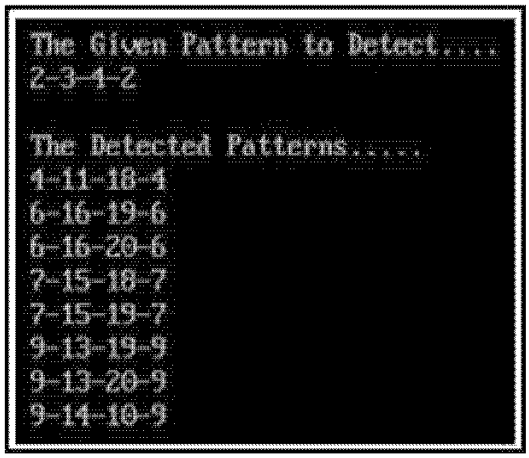

Fig. 12: Eight detected loop patterns

The algorithm was written in C++ and compiled with C++. The experiment was run on Intel Core I5-3230M CPU+2.60 GHz laptop with 4 GB memory running MS-Windows 7.

CONCLUSION

The researchers have proposed university graph as social attributed graph which compromises of various nodes having its attributes labelled with a type of work title such as Dean, Associate Dean, Professor and Associate Professor. From this proposed university attributed graph, the researchers have successfully detected the given line and loop patterns. For this detection of exact line and loop patterns or sub-graphs, the researchers have proposed an algorithm using graph mining techniques. Further, the proposed algorithm was implemented using C++ programming language and observed satisfactory results.

RECOMMENDATION

The future ressearch lies with detection of inexact patterns as well as some other patterns such as star and elongated star.

REFERENCES

Cook, D.J. and L.B. Holder, 2007. Mining Graph Data. John Wiley & Sons, Hoboken, New Jersey, USA.,.

Gomes, C.S., J.N. Amaral, J. Sander, J. Siu and L. Ding, 2014. Heavyweight pattern mining in attributed flow graphs. Proceedings of the IEEE International Conference on Data Mining (ICDM), December 14-17, 2014, IEEE, Shenzhen, China, ISBN:978-1-4799-4302-9, pp: 827-832.

Jorgensen, Z., T. Yu and G. Cormode, 2016. Publishing attributed social graphs with formal privacy guarantees. Proceedings of the 2016 International Conference on Management of Data, June 26-July 1, 2016, ACM, San Francisco, California, USA., ISBN:978-1-4503-3531-7, pp: 107-122.

Rao, B., A. Mitra and P. Padhi, 2015. An approach to detect common community sub-graph between two community graphs using graph mining techniques. Proceedings of the 6th International Conference on Recent Trends in Information, Telecommunication and Computing (ITC15), March 28, 2015, ITC, Chennai, India, pp: 177-185.

Rao, B., A. Mitra and U. Narayana, 2014. An approach to study properties and behavior of social community network using graph mining techniques. Proceedings of the International Conference on DIGNATE 2014:: ETEECT 2014 at India, September 25, 2014, India International Centre, New Delhi, India, ISBN: 978-93-5196-068-3, pp: 13-17.

Rao, B., H.S. Maharana and S.N. Mishra, 2016. An approach to detect sub-community graph in n-community graphs using graph mining techniques. Proceedings of the IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), December 15-17, 2016, IEEE, Chennai, India, ISBN:978-1-5090-0612-0, pp: 1-6.

Silva, A., W. Meira Jr. and M.J. Zaki, 2012. Mining attribute-structure correlated patterns in large attributed graphs. Proc. VLDB Endowment, 5: 466-477.

Tong, H., C. Faloutsos, B. Gallagher and T. Eliassi-Rad, 2007. Fast best-effort pattern matching in large attributed graphs. Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 12-15, 2007, ACM, San Jose, California, USA., ISBN: 978-1-59593-609-7, pp: 737-746.

Zhang, Q., X. Song, X. Shao, H. Zhao and R. Shibasaki, 2014. Attributed graph mining and matching: An attempt to define and extract soft attributed patterns. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, June 23-28, 2014, Columbus, Ohio, USA., ISBN:978-1-4799-5118-5, pp: 1394-1401.