Journal of Modern Technology and Engineering
Vol.4, No.3, 2019, pp.211-225

Jomard Publishing

A NEW SOLUTION METHOD FOR GRAPH CLUSTERING PROBLEM

Burak Ordin*, Duygu Selin Ballı

Department of Mathematics, Faculty of Science, Ege University, Izmir, Turkey

Abstract. The purpose of the graph clustering problem is to divide the nodes of a given graph into clusters according to their similarities/distances. Graph clustering problems is NP-hard. In addition, the modularity is a quality measurement metric used for graph clustering and the modularity is strongly NP-complete. Therefore, it is very important to develop approximate and heuristics algorithms for graph clustering. This paper presents a new approach called the incremental graph clustering problem. The proposed method uses a ratio for each node of the given graph. The ratios are calculated by the adjacency matrix of the given graph. The proposed incremental approach is based on greedy, so it is uses collected information from previous iterations. It is illustrated that efficiency of the approach with computational experiments on 14 real world data sets in the literature. The results obtained by the incremental graph clustering algorithm and the k-spanning tree algorithm in the literature on 14 datasets are evaluated using by modularity quality metric for graph clustering. The values of this metric show the efficiency of the proposed algorithm, especially in large-scale data sets.

Keywords: Graph clustering, data mining, clustering, greedy approach, incremental clustering.

Corresponding author: Burak Ordin, Department of Mathematics, Faculty of Science, Ege University, Izmir, Turkey, e-mail: *burak.ordin@ege.edu.tr*

Received: 19 June 2019; Accepted: 2 November 2019; Published: 25 December 2019.

1 Introduction

Data mining is a collection of methods developed to identify data that can be understood and useful from data warehouse, including many techniques such as clustering, classification, association analysis.

With rapidly evolving technology, the size of data accumulated in databases is rapidly increasing. The importance of discovering useful information from these growing databases, that is the importance of data mining, is an unignorable fact. There are various data mining methods that can be used for this discovery process. Clustering is one of the most commonly used data mining methods. The clustering problem is a process of dividing a set of data into sub-clusters based on similarities. It is expected that the intracluster similarity will be maximum and the intercluster similarity will be minimum at the end of the clustering process.

The evaluation of the results obtained by applying the clustering algorithms is also a very important step. There are various clustering quality measurement methods. The choice of the quality measurement method depends on the clustering algorithm, the similarity-distance metric, and the problem. According to all these factors, the selection of the quality measurement method also has a critical decision. In the article published by Schaeffer (2007), the problem of evaluating clustering algorithms is addressed. Both global and local approaches have been reviewed and delicate issues such as choosing the appropriate method for the current task, choosing good parameter values and evaluating the resulting clustering quality have been emphasized. One of the most important aspects of graph clustering is the evaluation of clustering quality. This is important not only for measuring the effectiveness of the clustering algorithm, but also for getting information about the dynamics of relations of network used as graph network's relations. In the article published by Almeida et al (2012), a new method has been proposed to assess the

internal density of a cluster. Due to the importance of quality measurements, there are many studies in the literature. Modularity known as one of the popular clustering index has been evaluated in the article published in 2007 (Brandes et al., 2007). Emmons et al.(2016) in their article, performed a rigorous analysis of four widely used network clustering algorithms. By means of this analysis, they have examined the relationship between stand-alone cluster quality metrics and information recovery metrics.

Graph clustering techniques are very useful for detecting heavily linked groups in a large graph. Data mining has a very wide application area and has found application for graph clustering. In the article published by Zhou et al. (2009), the SA-Cluster algorithm, a new graph clustering algorithm based on both structural and attribute similarities, was proposed. The method proposed by Zhou et al. (2009) partitions a large graph associated with attributes into k clusters so that each cluster contains a densely connected subgraph with homogeneous attribute values and when designing the method, the K-Medoids clustering approach is considered. In the paper published by Kulis et al (2009), a new algorithm SS-KERNEL-KMEANS was proposed to optimize a semisupervised clustering objective that can cluster both vector-based and graph-based data. Almost all of approaches that can be discover characteristic patterns from graphs-structured data are not suitable for such applications that require a complete search for all frequent subgraph patterns in the data. Inokuchi et al. (2003) proposed a novel priciple and its algorithm that derive the characteristic patterns which frequently appear in graph-structured data. While developing the algorithm, they have extended the Apriori algorithm, which is usually used in association rule extraction.

The graph theory can be applied to the study field of many different disciplines. The theory, which is used in a wide range of fields from sociology to computer science to business engineering to industrial engineering, aims at simply modeling a real-life problem with a graph (Seker, 2015). Structured information has been the source of many researches for data mining over the last decade. The bioinformatics is an important area of application in this context. In the paper published by Parthasarathy et al (2010), the main results in the area were investigated by examining the related algorithmic contributions and applicability.

There are two broad categories of graph clustering methods: local and global. However, since graph clustering problems fall under the category of NP-hard problems, practical solutions are derived using heuristics and approximation algorithms. Greedy approaches are successful for graph partitioning. Jain et al. (1998) have proposed some greedy algorithms for k-way graph partitioning and experimental results have presented for large sample of two types of randomly generated graphs and some large real world graphs.

The rest of this paper is organized as follows. In Section 2, we introduce the data clustering problem and its nonsmooth nonconvex optimization formulation. In the next section, we mention the graph clustering and its solution methods. The proposed incremental graph clustering algorithm is presented in Section 4. Section 5 is dedicated to the experimental results obtained by the the proposed method and the k-spanning tree algorithm. Finally, conclusions are discussed in Section 6.

2 Data clustering problem and its mathematical model

Data clustering is one of the most important methods of data mining, and is used in many areas to provide meaningful and useful information. In other words, a data set is divided into meaningful subsets in clustering. The need to produce meaningful and useful output is not only for data mining, but also for many areas such as pattern recognation, statistics, medicine, and marketing. For this reason, clustering analysis is an interdisciplinary field of study with applications in various fields.

In the data clustering problem, no label information is given. Here, the process of assigning the data to the subclusters in a meaningful manner is based on the similarity or distance measure

between data. Correspondingly, it is expected that the similarity between data in the same cluster is greater than the similarity between data in different clusters (Xu & Wunsch, 2005).

The mathematical model of the clustering problem is as follows:

Let A be a set of n dimensional m points in the space R^n such that $A = \left\{a^1, \ldots, a^m\right\}$, $i = 1, \ldots, m$ and $a_i \in R^n$. The purpose of the clustering problem is to divide the data in the set A into k clusters A^j for $j = 1, .., k$ satisfying the following conditions (Bagirov & Mardaneh, 2006):

1. $A^j \neq \emptyset$, j= $1, \ldots, k$;
2. $A^j \cap A^l$=$\emptyset$, j, l= $1, \ldots, k$, $j \neq l$;
3. A=$\bigcup_{j=1}^{j} A^j$.
4. There is no constraint on A^j clusters for j= $1, \ldots k$.

A^j , $j = 1, \ldots, k$ are called as clusters. Each A^j cluster can be represented $x^j \in R^n$ centers for $j = 1, \ldots, k$.

The distance between data points is calculated by a measure called "similarity measure". This measure is defined by the distance to the center of a data point. A clustering problem can be reduced to an optimization problem as follows:

Let $d(x, y)$ be the distance between the x and y points. Nonsmooth nonconvex optimization formulation of the clustering problem is as follows (Ordin & Bagirov, 2015):

$$\text{Min}: \ f_k(x) = \frac{1}{m} \sum_{i=1}^{m} \min_{j=1,\ldots,k} d(x^j, a^i), \tag{1}$$

subject to

$$x = \left(x^1, \ldots, x^k\right) \in R^{n \times k} \tag{2}$$

f_k are called as cluster functions.

Clustering algorithms can be divided into five classes in general: partitioned methods, hierarchical methods, density based methods, grid based methods and model based methods. In the partitioned clustering method, which is the clustering technique used in this article, n data points are partitioned into k clusters for $k \leq n$. Here, each data point belongs to only one cluster.

3 Solution methods for the graph clustering problem

Clustering is one of the most commonly used data mining techniques and most traditional clustering algorithms can be applied on structured data. However graph data is unstructured data. Nevertheless, many data types can be converted to graph data type and better solutions can be obtained using these converted data. In addition, traditional data clustering algorithms can be extended to fit the graph data and graph clustering has many applications such as computational biology, software bug localization and computer networking. Therefore, developing new approaches to solve the graph clustering problem is important (Aggarwal & Wang, 2010; Lee et al., 2002).

The graph clustering problems are typically defined into two categories: node clustering algorithms and graph clustering algorithms. Node clustering algorithms are generalizations of multi-dimensional clustering algorithms. The edges are associated with numerical values. These numerical values represent the similarity or distance between the concerned two nodes. These values are used in order to create clusters of nodes. The objective function of clustering may

either be minimized or maximized. Node clustering algorithms are useful for massive graphs which have a large number of nodes. Besides, there are a large number of graphs and these graphs are clustered as objects based on their structural behavior in the case of graph clustering. Using their structures is a challenging problem (Aggarwal & Wang, 2010; Flake et al., 2004).

The clustering algorithms are useful for graph data. Rattigan et al . (2017) have investigated two simple algorithms: the k-medoids algorithms for graph and the Girvan-Newman method based on edge betweenness centrality. They have shown that these algorithms are effective at graph clustering but their computational complexities are large for even moderately sized graphs. In the multi-way graph partitioning, it is aimed to partition a graph into $k > 2$ components, so that the total weight of edged whose ends lie in different partitions is minimized. Kerninghan-Lin algorithm is a well known algorithm for this problem. However, implementing this algorithm is difficult with increasing value of number of cluster because of time-complexity (Aggarwal & Wang, 2010). The complexity problem is frequently encountered in large graph data sets. Therefore, improving approximation algorithm is significant for graph clustering.

In the node clustering problem, it is aimed to partition the graph into k groups of nodes. There are various algorithms for this partition and a few algorithms are given below. The k-spanning tree algorithm aims to partition a set of nodes of a graph to k cluster. Firstly, the minimum spanning tree (MST) is found for the graph. To find the MST of a graph, we can use the Prim's algorithm or Kruskal algorithm. After finding the MST of the graph, $k-1$ edges from MST are removed to obtain k cluster (Samatova et al., 2013).

Algorithm: The k-spanning tree algorithm
Input: A weighted graph $G = (V, E)$
Output: A list of clusters
1. Obtain the minimum spanning tree of the input graph G
2. Remove $k-1$ edges from the minimum spanning tree
3.**return** Clusters

Shared nearest neighbor is a proximity measure and it denotes the number of neighbor nodes common between any given pair of nodes. Given an input graph G, weight (u, v) with the number of shared nearest neighbor between u and v. The Jarvis-Patrick algorithm is as follows (Samatova et al., 2013).

Algorithm: The Jarvis-Patrick algorithm
Input: A graph $G = (V, E)$
Input: Threshold τ
Output: A list of clusters
1.Obtain the shared nearest neighbor graph of the input graph G
2.Remove edges from the shared nearest neighbor graph of G with weights less than τ
3.**return** Clusters.

In graph theory, betweenness centrality measures the degree to which a vertex (or edge) occurs on the shortest path between all the other pairs of nodes. There are two types: vertex betweenness and edge betweenness. Vertices and edges with high betweenness form good starting points to identify clusters. Girvan and Newman algorithm is an edge-betweenness clustering algorithm. An input graph and a threshold μ are taken from user. The edge with the highest betweenness centrality is removed. When the highest betweenness centrality in the graph falls below μ, the algorithm stops (Samatova et al., 2013).

Algorithm: The Girvan and Newman algorithm
Input: A weighted or unweighted graph $G = (V, E)$
Input: Threshold μ
Output: A list of clusters
1.**while** $|E(G)| > 0$ **do**
2.$C_{u,v}$ –betweenness centrality of edge (u, v)
3.Calculate $C_{u,v}$ for all $(u, v)\ \epsilon\ E(G)$
4.$maxBetweennessEdge = (x, y)$:$C_{x,y}$ is minimum over all (x, y) in $E(G)$
5.$maxBetweennessEdgeValue = C_{x,y}$
6.**if** $maxbetweennessValues \geq \mu$ **then**
7.$E(G) = E(G) - \{maxBetweennessEdge\}$
8.**else**
9.Break out of loop
10.**end**
11.end
12.return Connected components of midefied G

In graph theory, a cut of a graph G $(Cu(G))$ is a set of edges whose removal disconnects the graph G. The minimum edge cut of a graph G $(M_{Cu}(G))$ is the cut with the minimum number of edges. The edge connectivity of a graph G $(EC(G))$ is the minimum number of edges whose removal will disconnect the graph G, i.e. $EC(G) = |M_{Cu}(G)|$. A graph $G = V, E$ with v vertices is highly connected if edge connectivity of G is greater than $\frac{v}{2}$. In accordance with these definitions, the highly connected subgraph algorithm is as follows (West, 2001):

Algorithm: The highly connected subgraph algorithm
Input: A weighted or unweighted graph $G = (V, E)$
Input: Threshold γ
Output: Highly connected component
1.**if** G is weighted **then**
2.$E_{remove} = \{(u, v)\ \epsilon\ E(G) : weight(u, v) < \gamma\}$
3.$E(G) = E(G) - E_{remove}$
4.G is now taken as unweighted
5.**end**
6.Remove self-loops from G
7.$HCS(G)$

Metrics are needed to measure the effectiveness of approximate algorithms. The choice of these metrics varies according to the problem and is critical. Emmons et al. (2016) have analyzed the metrics and algorithms for graph clustering. They have examined the relationship between stand-alone cluster quality metrics and information recovery metrics. In the paper of Emmons et al. (2016), the modularity, conductance and coverage have used as stand-alone quality metrics and the results have shown that the conductance is the best of stand-alone metrics.

In this paper, modularity is used as quality measurement metric for graph clustering. Modularity is a quality index for clusterings. Given a simple graph $G = (V,\ E)$ and let k be the number of cluster, $q(K)$ be the modularity of each cluster K in a clustering C is defined as

$$\mathrm{q(K)} = \sum_{\mathrm{C \in K}} \left[\frac{|\mathrm{E(C)}|}{\mathrm{m}} - \left(\frac{\sum_{\mathrm{v \in C}} \deg(\mathrm{v})}{2\mathrm{m}} \right)^2 \right] \tag{3}$$

where $\mathrm{E(C)}$ is the set of intra-cluster edges and $m = |E|$ (Brandes et al., 2007). In this formula, first term is the probability of intra-cluster edges in cluster K and second term is the probability of either and intra-cluster edge in cluster K or of an inter-cluster edge incidens on cluster K. Note that the first term $|E(C)|/m$ is also known as coverage (Brandes et al., 2007;

Emmons et al., 2016).

The modularity of a graph for a clustering C is given by (Emmons et al., 2016)

$$q\left(C\right)=\sum_{K\in C}q(K) \tag{4}$$

The modularity of a graph falls in the range 0 to 1, the optimal score is 1 (Emmons et al., 2016).

4 New approach for solution of the graph clustering problem

In the greedy approach, the choice (locally optimal) that will come closest to optimal result is made. Greedy algorithms developed for the solving of graph clustering problem in general can converge only to local minima and these local minima are significantly different from global solutions as the number of clusters increases. Therefore, the incremental algorithms have an important advantage for converging to the global minimum (Bagirov & Mardaneh, 2006).

In this section, a new approach is presented for graph clustering. The given algorithm is incremental, so it is aimed to find a global optimal solution with local optimal solutions. The obtained results by the algorithm is evaluated with modularity quality metric. This metric ranges from 0 to 1 and 1 is the optimal value. The results are presented in Table 3.

The proposed algorithm, the incremental graph clustering algorithm (IGC), is as follows:

Algorithm: The incremental graph clustering algorithm
Input: A weighted graph $G=(V,E)$
Input: The adjacency matrix of $G=(V,E)$
Output: A list of clusters
1. Set $p=1$
2. For each node, calculate the following ratio with the adjacency matrix
$\frac{\textit{the sum of weights of edges ending with the node}}{\textit{the number of edges ending with the node}}$
3.Set $t=1$
4.Assign the number of nodes that have non zero ratio to d and name these nodes as candidate centers
5.The smallest tth candidate center with no direct connection at preselected centers save as pth center y_p and go to Step 12
6.Set $t=t+1$
7.**if** $t>d$ **then**
8.go to Step 16
9. **else**
10. go to Step 5
11. **end**
12.Set the row and column values where the node selected as the pth center is 0 and set $p=p+1$
13. With the Floyd-Warshall algorithm, find the nearest center for each $v\in V$ and assign it to the corresponding cluster
14.Calculate the modularity for p cluster by equation (1) and assign $m_i=y_i$ for each $i\in\{1,2,\ldots,p\}$ if a better result is obtained than the previous modularity values
15. Go to Step 2
16. **return** Centers m_i for $i\in\{1,2,\ldots,p\}$ and the clusters to which each node belongs

The algorithm firstly calculates the ratio mentioned in Step 2 for each node with the help of the adjacency matrix. Nodes with a non-zero ratio are considered as candidate centers and the node with the smallest ratio providing the appropriate conditions is selected as the center. As long as candidate centers exist, the clustering continues by increasing the number of clusters. As the new center is found, with the help of the Floyd-Warshall algorithm, each node is assigned

to the cluster with the nearest center. For new clusters found, the modularity value given by equation (1) is calculated. A better result than the previous modularity values is founded, ie if a better clustering is obtained, the current clusters and the information for which cluster belongs to each node are stored. When the candidate center has no left, the clustering result with the best modularity value is printed. Besides, we evaluated the algorithm with modularity.

Given a weighted graph with positive or negative edges but with no negative cycles, the Floyd-Warshall algorithm finds shortest paths between all pairs of vertices in the graph (Cormen et al., 2009). Given a graph G with vertices V and edges E, the Floyd-Warshall algorithm can be expressed as (Floyd, 1962):

Algorithm: The Floyd-Warshall algorithm
Input: A weighted graph $G = (V, E)$
Output: An $|V| \times |V|$ distance matrix $D = dist[i, j]$ for $i, j \in \{1, 2, \ldots, |V|\}$
1. $dist\,[i, j] = \infty\ \forall ij\ \in V$
2. $dist\,[i, j] = 0\ \forall i \in V$
3. $dist\,[i, j] = \ w\,((i, j))\ \ \forall (i, j) \in E$
4. **for** $i = 1$ **to** $|V|$ **do**
5. **for** $j = 1$ **to** $|V|$ **do**
6. **for** $k = 1$ **to** $|V|$ **do**
7. **if** $dist\,[j, k] > dist\,[j, i] + dist[i, k]$
8. **then** $dist\,[j, k] = dist\,[j, i] + dist\,[i, k]$
9. **end**
10. **end**
11. **end**
12. **end**
13**return** the distance matrix D

where dist[i, j] is the minimum distances between the vertices i and j, and w ((i, j)) is the weight of the edge (i, j) ∈ E. The output of the algorithm is an |V| × |V| matrix dist such that dist[i, j] contains the length of a shortest path form vertex i and vertex j.

In order to explain the algorithm, we present a step-by-step study on the graph G = (V, E) given in Fig. 1.

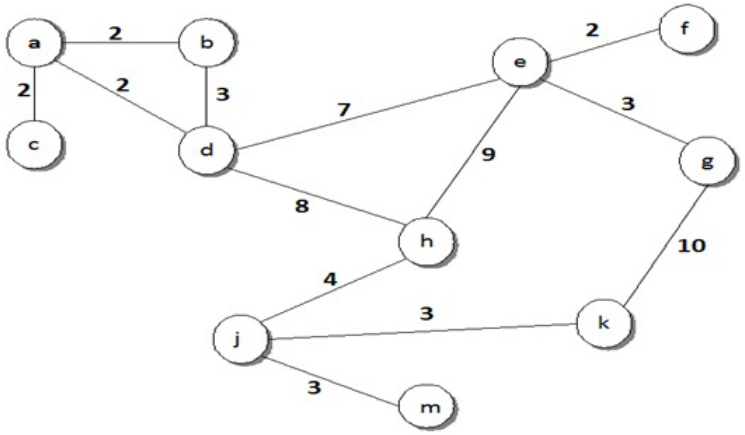

Figure 1: The graph $G = (V, E)$ with $|V| = 11$ and $|E| = 13$

Firstly, the 11×11 following matrix M is obtained with weigths of edges. In this matrix, rows and columns represent nodes.

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**j**	**k**	**m**
a	0	2	2	2	0	0	0	0	0	0	0
b	2	0	0	3	0	0	0	0	0	0	0
c	2	0	0	0	0	0	0	0	0	0	0
d	2	3	0	0	7	0	0	8	0	0	0
e	0	0	0	7	0	2	3	9	0	0	0
f	0	0	0	0	2	0	0	0	0	0	0
g	0	0	0	0	3	0	0	0	0	10	0
h	0	0	0	8	9	0	0	0	4	0	0
j	0	0	0	0	0	0	0	4	0	3	3
k	0	0	0	0	0	0	10	0	3	0	0
m	0	0	0	0	0	0	0	0	3	0	0

Using the matrix M, the ratios mentioned in Step 2 for each node are calculated and the following values are obtained.

a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**j**	**k**	**m**
2	2,5	2	5	5,25	2	6,5	7	3,33	6,5	3

According to these ratios, the node a, c and f have the smallest value. Therefore, we randomly select one of these 3 nodes. Suppose that the node a is selected so, our first center is the node a. For 1 center, the modularity value is calculated as 0. After the first center is determined, the corresponding row and column values of the selected node as center in the M matrix are set to 0 and we update the matrix M as following.

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**j**	**k**	**m**
a	0	0	0	0	0	0	0	0	0	0	0
b	0	0	0	3	0	0	0	0	0	0	0
c	0	0	0	0	0	0	0	0	0	0	0
d	0	3	0	0	7	0	0	8	0	0	0
e	0	0	0	7	0	2	3	9	0	0	0
f	0	0	0	0	2	0	0	0	0	0	0
g	0	0	0	0	3	0	0	0	0	10	0
h	0	0	0	8	9	0	0	0	4	0	0
j	0	0	0	0	0	0	0	4	0	3	3
k	0	0	0	0	0	0	10	0	3	0	0
m	0	0	0	0	0	0	0	0	3	0	0

Using the updated matrix M, the ratios mentioned in Step 2 for each node are calculated and the following values are obtained.

a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**j**	**k**	**m**
0	3	0	6	5,25	2	6,5	7	3,33	6,5	3

According to these ratios, the node f has the smallest value, so we select the node f as second center. We assign the nodes that is not center to clusters using the Floyd-Warshall algorithm and calculate the modularity as 0.269230. After the second center is determined, the corresponding row and column values of the node f in the M matrix are set to 0 and we update the matrix M as following.

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**j**	**k**	**m**
a	0	0	0	0	0	0	0	0	0	0	0
b	0	0	0	3	0	0	0	0	0	0	0
c	0	0	0	0	0	0	0	0	0	0	0
d	0	3	0	0	7	0	0	8	0	0	0
e	0	0	0	7	0	0	3	9	0	0	0
f	0	0	0	0	0	0	0	0	0	0	0
g	0	0	0	0	3	0	0	0	0	10	0
h	0	0	0	8	9	0	0	0	4	0	0
j	0	0	0	0	0	0	0	4	0	3	3
k	0	0	0	0	0	0	10	0	3	0	0
m	0	0	0	0	0	0	0	0	3	0	0

We calculate the ratios for each node in Step 2 using the new matrix M.

a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**j**	**k**	**m**
0	3	0	6	6,33	0	6,5	7	3,33	6,5	3

The node b and m have the smallest ratio. The node b is directly connected to the preselected center, so we select the node m as the third center. We assign the nodes that is not center to clusters using the Floyd-Warshall algorithm and calculate the modularity value as 0.43934. After the third center is determined, the corresponding row and column values of the selected node as center in the M matrix are set to 0 and we update the matrix M as following.

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**j**	**k**	**m**
a	0	0	0	0	0	0	0	0	0	0	0
b	0	0	0	3	0	0	0	0	0	0	0
c	0	0	0	0	0	0	0	0	0	0	0
d	0	3	0	0	7	0	0	8	0	0	0
e	0	0	0	7	0	0	3	9	0	0	0
f	0	0	0	0	0	0	0	0	0	0	0
g	0	0	0	0	3	0	0	0	0	10	0
h	0	0	0	8	9	0	0	0	4	0	0
j	0	0	0	0	0	0	0	4	0	3	0
k	0	0	0	0	0	0	10	0	3	0	0
m	0	0	0	0	0	0	0	0	0	0	0

Using the new matrix M, the ratios mentioned in Step 2 for each node are calculated and the following values are obtained.

a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**j**	**k**	**m**
0	3	0	6	6,33	0	6,5	7	3,5	6,5	0

The node that have the smallest ratio such that is not directly connected to preselected centers is the node g. Therefore, we select the node g as the fourth center. We assign the nodes that is not center to clusters using the Floyd-Warshall algorithm and calculate the modularity value as 0.423076. After the fourth center is determined, the corresponding row and column values of the selected node as center in the M matrix are set to 0 and we update the matrix M as following.

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**j**	**k**	**m**
a	0	0	0	0	0	0	0	0	0	0	0
b	0	0	0	3	0	0	0	0	0	0	0
c	0	0	0	0	0	0	0	0	0	0	0
d	0	3	0	0	7	0	0	8	0	0	0
e	0	0	0	7	0	0	0	9	0	0	0
f	0	0	0	0	0	0	0	0	0	0	0
g	0	0	0	0	0	0	0	0	0	0	0
h	0	0	0	8	9	0	0	0	4	0	0
j	0	0	0	0	0	0	0	4	0	3	0
k	0	0	0	0	0	0	0	0	3	0	0
m	0	0	0	0	0	0	0	0	0	0	0

Using the updated matrix M, the ratios mentioned in Step 2 for each node are calculated and the following values are obtained.

a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**j**	**k**	**m**
0	3	0	6	8	0	0	7	3,5	3	0

We select the node h that have the smallest ratio such that is not directly connected to preselected centers as fifth center. We assign the nodes that is not center to clusters using the Floyd-Warshall algorithm and calculate the modularity value as 0.281065. After the fifth center is determined, the corresponding row and column values of the selected node as center in the M matrix are set to 0 and we update the matrix M as following.

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**j**	**k**	**m**
a	0	0	0	0	0	0	0	0	0	0	0
b	0	0	0	3	0	0	0	0	0	0	0
c	0	0	0	0	0	0	0	0	0	0	0
d	0	3	0	0	7	0	0	0	0	0	0
e	0	0	0	7	0	0	0	0	0	0	0
f	0	0	0	0	0	0	0	0	0	0	0
g	0	0	0	0	0	0	0	0	0	0	0
h	0	0	0	0	0	0	0	0	0	0	0
j	0	0	0	0	0	0	0	0	0	3	0
k	0	0	0	0	0	0	0	0	3	0	0
m	0	0	0	0	0	0	0	0	0	0	0

We calculate the ratios in Step 2 as following.

a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**j**	**k**	**m**
0	3	0	5	7	0	0	0	3	3	0

The all nodes that not selected as center are directly connected to preselected centers. Therefore, we cannot select any node as sixth center and the algorithm stops the searching a new center.

Finally, we store the centers that give the best modularity value. When we look the modularity value, the best clustering is for 3 clusters. As a result, the algorithm returns these centers and the clusters to which each node belongs. We visualize the result of the algorithm in Fig.2.

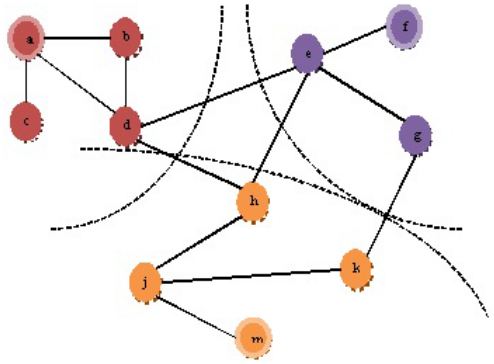

Figure 2: The resulting clustering of $G = (V, E)$ with the IGC algorithm

5 Experimental results

To verify the efficiency of the proposed algorithm, numerical experiments with 14 real-world data sets have been carried out on a PC Intel(R) Core(TM) i3 with CPU 2.13 GHz and RAM 3 GB running under Windows 7. We used 14 problems from OR Library (Beasley, 1990), ZIB (Zuse Institute Berlin, 2017) and Algorithms (Sedgewick & Wayne, 2011) to test methods. Information about datasets are given in Table 1. The first 6 data sets in Table 1 are taken from ZIB. The next 6 data sets are taken from OR Library, and the last two data sets are taken from Algorithms.

Table 1: Information about data sets

The name of data set	The number of vertices	The number of edges
cb450.30.6.47	30	47
cb450.45.8.98	45	98
cb450.47.8.99	47	99
cb450.47.9.101	47	101
cb450.61.9.187	61	187
cb512.61.8.187	61	187
steinb1	50	63
steinb4	50	100
steinb7	75	94
steinb10	75	150
steinb14	100	125
steinb17	100	200
tinyEWG	8	16
mediumEWG	250	1273

We have evaluated the IGC algorithm and the k-spanning algorithm with modularity that is discussed in Section 2. More clearly, we use the modularity as an objective function. This metric is between 0 and 1. 1 is the optimal value, that is, the higher the value, the better the clustering result. For all data sets, the cluster number k is considered as the number of nodes |V| that is, the algorithm continues to search a new center until the new center can not be found. The output of the algorithm is determined by the best modularity value. We have presented the results of the IGC algorithm for all clustering for the dataset cb450.30.6.47. The maximum cluster number for this dataset was 18 and so, clustering has performed to 18. For each clustering, modularity value has calculated and presented in the Table 3 and Fig.3.

When looking at the Table 2 and the Fig. 3, the best modularity value for the problem came for 6 clusters. Therefore, the output of the algorithm is results of clustering for 6 clusters. We have made this comparisons for all datasets and the clustering which in is obtained the best modularity values, have stored. Besides, we use the k-spanning tree algorithm mentinoned in Section 3 to compare the algorithm. Computational experiments for the k-spanning tree

Table 2: Obtained modularity values by the IGC algorithm for each clustering for the dataset cb450.30.6.67

The number of clusters	Modularity
1	0
2	0.138977
3	0.152784
4	0.211634
5	0.180285
6	0.284065
7	0.261204
8	0.222612
9	0.225215
10	0.193979
11	0.173495
12	0.168062
13	0.155727
14	0.144183
15	0.131621
16	0.114192
17	0.070733
18	0.119737

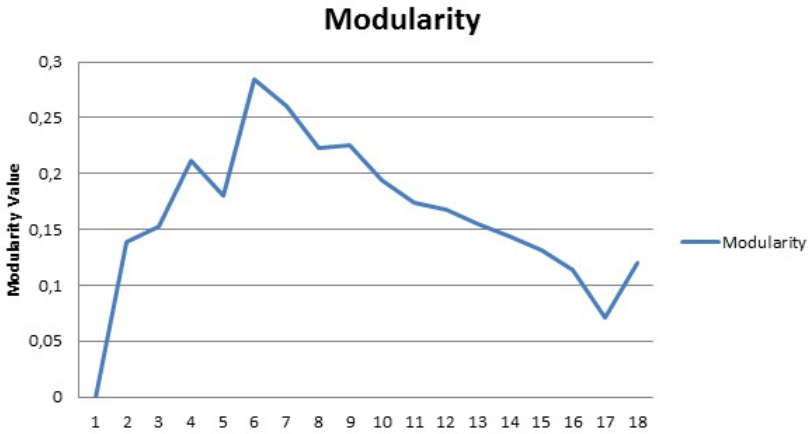

Figure 3: The relation between modularity values for the IGC algorithm and the number of clusters for the dataset cb450.30.6.67

algorithm on 14 real world datasets have been performed and the modularity quality metric is calculated. Modularity values calculated by both algorithms have been used for comparison.

We make comparison between the k-spanning tree algorithm and the IGC algorithm by the following rate:

$$improvement\ rate = \frac{\triangle_{IGC} - \triangle_{k-spanning}}{\triangle_{k-spanning}} \times 100 \tag{5}$$

Here, $\triangle_{\mathrm{IGC}}$ is the modularity value obtained by the IGC algorithm and $\triangle_{\mathrm{k-spanning}}$ is the modularity value obtained by the k-spanning algorithm. Positive improvement rate shows that the IGC algorithm works better than the k-spanning algorithm on related problem. If the improvement rate is negative, this means that the results obtained with the k-spanning tree algorithm are better than results obtained with the IGC algorithm on related problem. The results of experiments on datasets in Table 1 are presented in the Table 3. Moreover we visualize the results by graphics in Fig.4 and Fig. 5.

Table 3: Obtained results with the IGC and k-spanning algorithm

Name of dataset	The number of clusters	The modularity value for the k-spanning tree algorithm	Runtime for the k-spanning tree algorithm (s)	The modularity value for the IGC algorithm	Runtime for the IGC algorithm (s)	Improvement rate (%)
cb450.30.6.47	6	0.3250	1	0.4627	9.428	42.36923
cb450.45.8.98	15	0.3711	1	0.4138	2.278	11.50633
cb450.47.8.99	11	0.4228	1	0.4424	2.004	4.635762
cb450.47.9.101	4	0.4687	1	0.4925	1.670	5.077875
cb450.61.9.187	9	0.3739	1	0.4158	1.784	11.2062
cb512.61.8.187	10	0.3648	1	0.4256	1.596	16.66667
steinb1	10	0.2706	1	0.6158	1.377	127.5684
steinb4	6	0.4353	1	0.4611	1.468	5.926947
steinb7	13	0.4083	2	0.6587	2.603	61.32746
steinb10	12	0.2547	2	0.4716	2.061	85.15901
steinb14	9	0.2321	2	0.6501	3.636	180.0948
steinb17	10	0.2563	2	0.4923	3.319	92.07959
tinyEWG	2	0.4063	1	0.3984	1.490	-1.94438
mediumEWG	16	0.1308	6	0.7177	30.719	448.7003

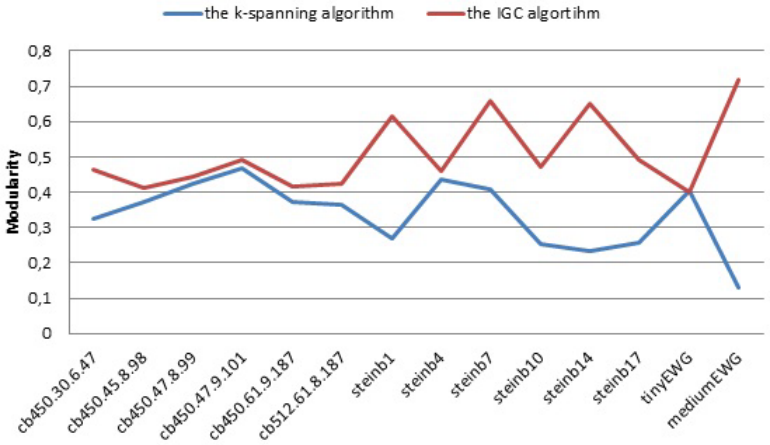

Figure 4: The graphic of the modularity values that obtained by the k-spanning and the IGC algortihm for each problem

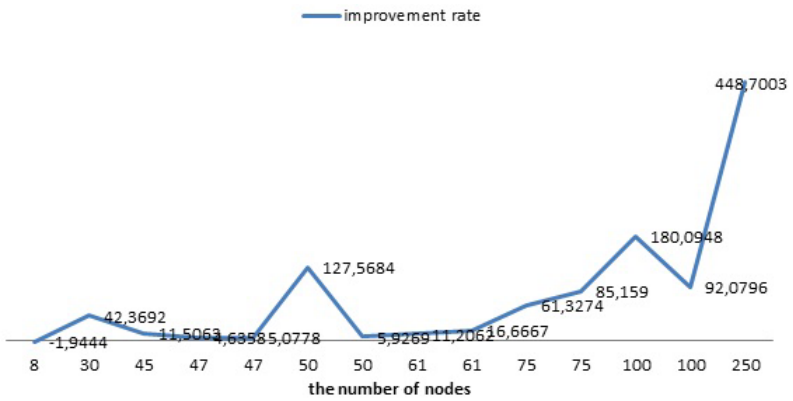

Figure 5: The relation between improvement rate and the number of nodes

The results obtained from experiments show the effectiveness of the IGC algorithm. It can

be seen from the Table 3 and Fig. 4 that in all datasets except the tinyEWG dataset, the IGC algorithm has better modularity values than the k-spanning algorithm. So, when modularity values are analyzed, it appears that the IGC algorithm works better, especially in large datasets. Looking at the run time of the algorithm, it seems reasonable.

Obviously, as the number of nodes increases, the IGC algorithm gives better results that the k-spanning tree algorithm. This conclusion is visualized in Fig. 3.

6 Conclusion

The graph theory is an interdisciplinary field. The theory, which is used in a wide range of fields aims at simply modeling a real-life problem with a graph. Graph clustering techniques are very useful for solving such problems. However, as the size of the problem increases, it becomes difficult to find an optimal solution. Therefore, it is important to develop approximate algorithms for such problems.

The main purpose of this work is to develop an incremental algorithm for graph clustering. In order to analyze the behavior of the new proposed approach on graph clustering, computational experiments have been carried out on 14 real world data sets in the literature for both the k-spanning tree and proposed algorithm. Modularity quality metric has been used to evaluate the algorithms. The values of modularity show the efficiency of the proposed algorithm.

References

Aggarwal, C.C., & Wang, H. (2010). A survey of clustering algorithms for graph data. *In Managing and mining graph data*, (pp. 275-301), Springer, Boston, MA.

Almeida, H., Neto, D.G., Meira Jr,W., & Zaki, M.J. (2012). Towards a better quality metric for graph cluster evaluation.*Journal of Information and Data Management*, *3*(3), 378.

Bagirov, A.M., & Mardaneh, K. (2006, December). Modified global k-means algorithm for clustering in gene expression data sets. In *Proceedings of the 2006 workshop on Intelligent systems for bioinformatics-Volume 73* (pp. 23-28). Australian Computer Society, Inc.

Beasley, J.E. (1990). OR-Library: distributing test problems by electronic mail. *Journal of the operational research society, 41*(11), 1069-1072.

Brandes, U., Delling, D., Gaertler, M., Görke, R., Hoefer, M., Nikoloski, Z., & Wagner, D. (2007, June). On finding graph clusterings with maximum modularity. In *International Workshop on Graph-Theoretic Concepts in Computer Science* (pp. 121-132). Springer, Berlin, Heidelberg.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., & Stein, C. (2009). Introduction to algorithms (3rd ed.). MIT pres. See in particular Section 25.2, *The Floyd-Warshall algorithm*, 693-695.

Emmons, S., Kobourov, S., Gallant, M., & Börner, K. (2016). Analysis of network clustering algorithms and cluster quality metrics at scale. *Plos one, 11*(7), e0159161.

Flake, G. W., Tarjan, R. E., & Tsioutsiouliklis, K. (2004). Graph clustering and minimum cut trees. *Internet Mathematics, 1*(4), 385-408.

Floyd, R.W. (1962). Algorithm 97: shortest path. *Communications of the ACM, 5*(6), 345.

Inokuchi, A., Washio, T., & Motoda, H. (2003). Complete mining of frequent patterns from graphs: Mining graph data. *Machine Learning, 50*(3), 321-354.

Jain, S., Swamy, C., & Balaji, K. (1998). Greedy algorithms for k-way graph partitioning. In *the 6th international conference on advanced computing* (p.100).

Kulis, B., Basu, S., Dhillon, I., & Mooney, R. (2009). Semi-supervised graph clustering: a kernel approach. *Machine learning, 74*(1), 1-22.

Lee, M.L., Yang, L.H., Hsu, W., & Yang, X. (2002, November). XClust: clustering XML schemas for effective integration. In *Proceedings of the eleventh international conference on Information and knowledge management* (pp. 292-299). ACM.

Ordin, B., & Bagirov, A.M. (2015). A heuristic algorithm for solving the minimum sum-of-squares clustering problems. *Journal of Global Optimization, 61*(2), 341-361.

Parthasarathy, S., Tatikonda, S., & Ucar, D. (2010). A survey of graph mining techniques for biological datasets. In *Managing and mining graph data* (pp. 547-580). Springer, Boston, MA.

Rattigan, M. J., Maier, M., & Jensen, D. (2007, June). Graph clustering with network structure indices. In *Proceedings of the 24th international conference on Machine learning* (pp. 783-790). ACM.

Samatova, N.F., Hendrix, W., Jenkins, J., Padmanabhan, K., & Chakraborty, A. (Eds.). (2013). *Practical graph mining with R.* CRC Press.

Schaeffer, S.E. (2007). Graph clustering. *Computer science review, 1*(1), 27-64.

Sedgewick, R., & Wayne, K. (2011). Algorithms. Addison-Wesley Professional.

Seker, S.E. (2015). Çizge Teorisi (Graph Theory). YBS Ansiklopedi, *2*(2), 17-29.

Zuse Institute Berlin. (21 September 2017). ZIB. July 2018. http://elib.zib.de/

West, D.B. others. (2001). Introduction to graph theory. Vol. 2.

Xu, R., & Wunsch, D. C. (2005). Survey of clustering algorithms.*IEEE Transactions on Neural Networks, 16*(3), 645-678.

Zhou, Y., Cheng, H., & Yu, J.X. (2009). Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment, 2*(1), 718-729.