Applied Mathematics & Information Sciences Letters
*An International Journal*

http://dx.doi.org/10.18576/amisl/040206

# Clustering Uncertain Graph Data Stream

*Zahra Varaminy Bahnemiry*[1,*], *Mir Mohsen Pedram*[2] *and Mitra Mirzarezaee*[3]

[1] Department of computer engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran
[2] Department of computer engineering, Faculty of Engineering, Kharazmi University, Tehran, Iran
[3] Department of computer engineering,Faculty of Computer Engineering,Science and Research Branch, Islamic Azad University, Tehran, Iran

Received: 23 Aug. 2015, Revised: 15 Nov. 2015, Accepted: 16 Nov. 2015
Published online: 1 May 2016

**Abstract:** Today, according to the increasing spread of information which people deal with, taking advantage of methods such as data mining to extract hidden knowledge from data is inevitable. Due to the extremely high volume of data in many applications and higher importance of new data, storage of these data is not effective in cost, so clustering these data is more important because of the data that are processed are always changing dynamically. Another problem in data mining is the issue of clustering of graph data stream. According to a number of existing algorithms for graph data stream clustering, choosing an appropriate algorithm has been challenged, which its challenge is time and space complexity. On the other hand, the uncertainty in edge graph stream should be taken into consideration to ensure reliable results that have not been investigated in studies so far. In this paper, a novel algorithm, for clustering of graph data stream considering uncertainty is investigated in a dynamic environment. Generally, the main innovation of this paper is to provide an approach for clustering uncertain graph data stream possessing a concept drift and dynamic. The results of the experiments conducted in this paper indicate the suitability of the proposed approach to this problem. Its time and space complexity is also reasonable.

**Keywords:** Data Mining, Clustering, Data Stream, Graph Data Stream, Uncertainty, Graph

## 1 Introduction

In the last recent years, creating graph databases (such as social networks and bank transactions) has caused data mining in graphs, or graph mining, to be considered a lot. Graphs and sub-graphs are data structures used in complicated object modeling. Graph is the main model for data representation and it is used in so many fields like chemistry informatics, biology informatics, social networks, bank transactions, etc. A lot of researches have been done on graphs and graph clustering is one of the most important issues among them. Graph clustering is mentioned in two different definitions: Node Clustering Algorithm and Structural Clustering Algorithm.[1] One of the important issues in graph clustering is the clustering of graph streams. If the data in a problem is data stream, no processing will be done after the storage, because there will be a huge load of data which is changing continuously, and also the new data have new groups that did not exist in the old one. Data processing after the storage is not possible due to the continuous growth of database, therefore the data streams must be processed while receiving, so that the data load will not cause any problems and the dynamics in the groups of data will be considered as well. Since the graph can be very large, it needs to be partitioned and stored on multiple machines in that case. If we have an upper bound on the cluster size (called Maximum cluster size), we will be able to do the distributed storage. No cluster will be too big to fit in one machine.

Some examples of graph data streams whose their clustering is considered a lot is as follows:

1. **Protein-Protein Interaction Networks**, in these networks, relations and interactions among proteins are represented as a graph. According to researches done on these networks, those proteins having the most interactions are classified in one group.
2. **Social Networks**, in the networks created from the interactions among living creatures, specially human beings, the relation between entities are described as links in a graph, such as relations of people in the

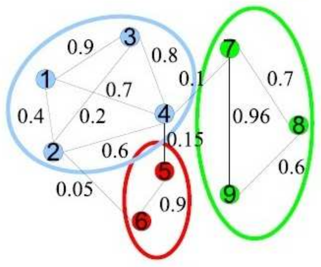
* Corresponding author e-mail: z.varaminy@srbiau.ac.ir

Internet and virtual networks, their interaction in bank and financial communications and their connection in broadcasting a rumor.

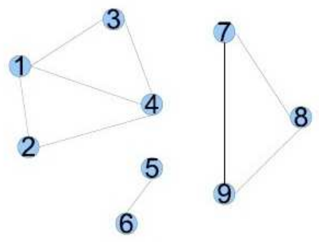
Clustering the mentioned networks expresses the groups that have the largest number of relations, like the groups of people broadcasting a rumor, or detecting the groups of financial interactions that involve money laundering, or the groups that broadcast a virus in a computer network or in a human society. On the other hand, each edge of the graph in this modeling denotes the relation between two entities and the probability of that edge denotes the intensity or strength of this relation. A social network is created from people (or organizations) that form the nodes. These nodes are linked to each other by some special sorts of interdependencies like friendship, kinship, common interests, common beliefs, etc. Analysis of social networks expresses the social relationships which are modeled by graph theory.

On the other hand, uncertainty in graph data is not much considered, specially no such research has been done on graph data streams. For instance, in studying bank transactions like money laundering, the exact amount of exchanges is not reported. Hence the edge corresponded to these transactions are presented with a probability value that involves uncertainty. In addition ,large scale data are imprecise and developers cannot be completely confident that data about individuals, or the connections between them, is accurate. For example, data collected through automated sensors [16], anonymized communication data (e.g. e-mail headers [16]), and self-reporting/logging on Internet scale networks [16] as a proxy for real relationships and interactions causes some uncertainty. As those studying and utilizing social networks have moved to enormous scales, they have frequently lost some accuracy.in wild and uncontrolled environments such as the Internet, biases can develop due to application design (e.g. default friends on MySpace) and malicious individuals (e.g. spammers building network connections in some automated way). The result of this noise is the introduction of tremendous levels of uncertainty in the data[16]. It is real-life examples to illustrate the applications of the proposed method. An obvious approach is to convert clustering uncertain graph data stream problem into the deterministic scenario by using edge probabilities as edge weights[19]. we are the first to formulate the clustering uncertain graph data stream problem, as it relates to connectivity issues in the presence of uncertainty [19]. In this paper, we take into account the reliability of graph and construct an uncertain graph data stream network, in which the reliability of each interaction is represented as a probability that are calculated based on the number of common neighbors of two nodes [4].

Let $G =< V, E, P >$ be a uncertain network, where V is a set of vertices, E a set of edges and P a set of probabilities of edge existence and presents the probability of the vertices are connected in a graph. The probability value is between 0 and 1 interval . In each cluster, there is a constraint on the maximum number of vertices. Since the graph can be very large, it needs to be partitioned and stored on multiple machines in that case. If we have an upper bound on the cluster size, we will be able to do the distributed storage. No cluster will be too big to fit in one machine. The aim is to detect all the clusters and partition the vertices V into clusters C1, C2, . . . , Ck, so that the sum probabilities values of the inter-cluster edges is minimized. We want to cluster the graph then the cost of clustering must be lowest that is sum probabilities value of nodes in different clusters or sum probabilities value of the inter-cluster edges . The offline algorithms are inefficient in an online or streaming fashion, because in the offline version it is considered that there is the entire graph at the first and also they are not incremental in nature and they are not designed to handle massive inputs. In online and dynamic setting, the graph may change rapidly with time due to additions and deletions of vertices and edges. offline graph clustering methods, like METIS, are mostly insensitive to the clustering evolution, because they usually ignore the emerging clusters and not designed for stream applications. Then we can compare new algorithm with another online algorithm [2]. Fig.1 shows example of uncertain graph stream we want to cluster it such as result of the clustering be correctly.



(a) Uncertain graph stream



(b) UGSC clustering

**Fig. 1:** Uncertain graph stream and result of UGSC clustering

Our improvements can be summarized as follows:

1.We propose a new clustering algorithm for an uncertain graph data stream. There are edge insertions and deletions in this algorithm.
2.We quantify the clustering *stableness* of an uncertain graph data stream. We compare the performance of algorithm in two data sets.
3.We show *weighted cut* (the total probabilities value of the inter-cluster) quality and throughput (number of edge insertions or deletions is handled per unit time[2]) experiments to compare Uncertain Graph Stream Clustering algorithm with the offline one.
4.We implement Uncertain Graph Stream Clustering algorithm on the Dblp and Youtube graphs stream.

This paper is organized as follows. The related work is discussed in section 2. The algorithm of the proposed method named Uncertain Graph Stream Clustering (UGSC) is presented in section 3. After that in Section 4, we present the experimental results. Finally, conclusion and future works is presented in Section 5.

## 2 Related Work

In recent years, many researches have been done on graph clustering. METIS offline clustering algorithm is a high quality algorithm[13] but it is not designed for clustering graph data stream. Aggarwal presented an algorithm for online graph data stream. Since this algorithm does not cover edge deletion, it is not usable for graph data stream clustering in sliding windows [5]. Stanton and Kliot designed some intelligent algorithms for the partitioning of graph data streams to vertices assuming that the graph is stored on the disk previously [10]. In [13], in order to explore communities in evolving networks, Kawadia presented a metric named Estrangement, but some changes in clusters have been neglected in very dynamic networks. In [15], Lin presented a framework for analyzing the communities and changes in them. Gupta presented a clustering method for analyzing the biological networks, but the algorithm was offline [12]. Bahmani presented an algorithm for finding the dense components of a graph by using a stream model in which it is assumed that all vertices in the graph is known and each edge is checked [3]. In [1], Angel presented an algorithm for maintaining the dense sub-graphs in which it is assumed that the graph is complete. Agrawal presented an algorithm for exploring the dense clusters in very dynamic graphs, but the clustering issue is different from finding the dense sub-graphs of a graph [11]. In [2], Eldawy suggested EIC algorithm for graph data stream clustering, but the presented algorithm is not sensitive to the evolution in clustering issue.
Recently some methods have been suggested for graph clustering that are suitable for static data and cannot be used for graph data streams. Furthermore, these methods are not appropriate for huge graphs. On the other hand, some algorithms have been suggested for clustering the graphs with uncertainty, however they are not suitable for clustering the uncertain graph data streams in dynamic and concept drift environments. Then there is clustering algorithms only for uncertain graph and also only for stream graph that means there is no clustering algorithm for combining stream and uncertain graph. So, there is no algorithm for clustering uncertain graph data stream. This paper handles the clustering problem for graph data stream that is uncertain, too. So In this paper a new algorithm is presented for this issue.

## 3 Proposed Method

In this section, we present UGSC algorithm for a windowed uncertain graph data stream.

### 3.1 System Architecture

*Window Manager* (WM) and *Graph Manager* (GM) are used in UGSC Algorithm[13]. (Fig.2).

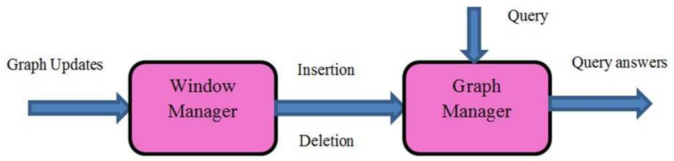


**Fig. 2:** window manager (WM) and graph manager (GM) are used in UGSC Algorithm[13].

### 3.2 Window Manager

For processing the large amount of data, streaming applications use a sliding or tumbling window. for example, in ‘*count-based tumbling window of one days*’ we save all events within the last one day. In this case, when the window is full, all events are deleted from the window and a new window starts. In ‘*time-based sliding window of 100k items*’ we save all events that number of items reaches 100k. In this case, when the window is full, the oldest event must be deleted from the window and new dynamic data replace it. These windows maintain, only the most recent updates of a graph, like the graph consisting of the last 2 million edges. As new updates continue to stream in, old updates are removed from the window. [2]. Fig.3 shows an example of a ‘*time-based sliding window of three days*’. In UGSC algorithm, we implemented both sliding window and tumbling window. Sliding window is used to test system performance under edges insertions and deletions because these are supported by sliding windows. When the window is full in tumbling window we must delete all the data in it so when we want to calculate the *throughput* by tumbling window the data are missed. It is problem we must use Sliding window for *throughput*. For stableness experiments, we use a tumbling window. In additions, for quality experiments we used a tumbling window.

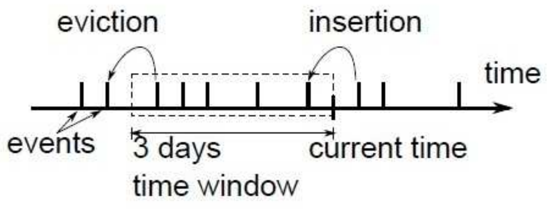


**Fig. 3:** A sample sliding window of three days[2].

## 3.3 Graph Manager

Graph Manager is the main data structures for saving current sampled update and clustering of the graph. Using the graph manager, we can answer all of queries in networks. The main query is: Which of the vertices place in the same cluster?

Table 1 and Table 2 show the two important data structures for graph manager in UGSC algorithm. As described in [13] the *Cluster Table* is a hash table for storing all the edges in a cluster and *Vertex Table* is a hash table too. This table is for storing vertices for mapping of a vertex ID to a cluster ID. In every cluster the ID of the first vertex indicates the cluster ID. *Cluster Table* is different from that was used in [13]. The first element in *Cluster Table* indicates *Cluster ID*, the second element is the *Cluster Size*, which indicates the number of vertices in the cluster. When the number of vertices in the cluster is bigger than maximum cluster size then we have constraint violation, but the third element is *Average Edges*, that shows the average probabilities in every cluster, the fourth is *Expected Density* that evaluate the density degree of a cluster, these two elements are new structures in UGSC algorithm that were not used in [13], the last element is a list of edges, saving all the edges in every cluster, This list consist of four elements: the timestamp of the edge, two vertices of the edge and probability of this edges. This list is sorted by the timestamps. In the add operations, an edge appends to the end of list and an edge remove from the front of list in delete operation.

Since after the deletion of an edge, we do not know if the other edges still forms a connected component. While there is an online method to keep track of connected

**Table 1:** vertex table sample [13].

| VertexID | ClusterID |
|---|---|
| 77 | 12 |
| .<br>.<br>. | .<br>.<br>. |

**Table 2:** cluster table sample.

| Cluster ID | Cluster Size | Average Edges | Expected Density | Edges:<time,v1,v2,probability> |
|---|---|---|---|---|
| 11 | 2 | 0.45 | 0.9 | <1,11,13,0.4>,<2,11,14,0.5> |
| .<br>. | .<br>. | .<br>. | .<br>. | .<br>. |

components [13], the algorithm is complicated to implement and expensive to maintain. Instead, we use a very simple method for edge deletion, as described in [2]. Upon a deletion of an edge from a cluster, we delete the entire cluster and then reinsert all the edges except for the deleted one. In this way, the insertion routine automatically merges connected components.

## 3.4 The Algorithm

In first step we create uncertain graph data stream, based on the uncertainty theory.

**Definition 1.** A uncertain graph data stream is defined as UGDS= (V, E, P), where P(E=$e_i$)=$p_i$, i=1, 2...m, P is a probability function expressing the intensity or strength of relation between two entities and defined as follows [4]:

$$P_i = \frac{(number\ of\ common\ neighbors\ of\ the\ two\ vertices)}{(minimum\ degree\ of\ the\ two\ vertices\ minus\ 1)} \tag{1}$$

Edge Insertion Algorithm is used to insert a new edge to the graph manager. Edge Deletion Algorithm shows the deletion algorithm in graph manager. For insertion, we test the constraint violation for the maximum cluster size, If the constraint is violated, we remove the oldest edges and reinsert rest of edges in the corresponding cluster with Edge Deletion Algorithm. In Edge Insertion Algorithm the threshold and $\varepsilon$ are inputs. There are some kinds of cases in this algorithm:

1. If both vertices of the edge are new, we check if probability value is bigger than *threshold* then we create new cluster.
2. If only one vertex of the edge is new, we check if *Expected Density* with new edge in cluster is bigger than *Expected Density* without this edge in cluster then we add the vertex in the cluster and update *Vertex Table* and *Cluster Table*, but if *Expected Density* with new edge in cluster is smaller than *Expected Density* without this edge in cluster then we need to compare *Clustering Similarity* inside this cluster and *Edge Similarity* with this cluster.

3. If the two vertices are in the same cluster then we compare *Clustering Similarity* inside this cluster without new edge and *Edge Similarity* with this cluster.
4. If the two vertices are in the different clusters then we compare two situations:
   - *Clustering Similarity* inside first cluster and *Edge Similarity* with first cluster
   - *Clustering Similarity* inside second cluster and *Edge Similarity* with second cluster

In this algorithm we use *sliding window*, this means if window is full we remove the oldest edge of window. In addition, the threshold and $\varepsilon$ are user defined. In the experiments, we set threshold = 0.4 and $\varepsilon$ = 0.02. In Edge Deletion algorithm, with removing edges, vertices may not belong to the same cluster as before and assign to a new cluster, so after the deletion of edge we reinsert rest of edges in corresponding cluster. This operation causes that we capture evolution in this algorithm. In Edge Deletion algorithm, after delete operation we will not cause constraint violations because there have not been constraint violations before the deletion.

## *3.5 Computing Expected Density, Clustering Similarity and Edge Similarity*

The main innovation of this paper is to provide an approach for clustering uncertain graph data stream and we used from combining definitions were expressed in [4], [6] and introduced a new formula and new algorithm for UGDS algorithm.

Using a describe in [4] *Expected Density* of Uncertain Graph Data Stream (UGDS) in a cluster is defined as follows:

**Definition 2.** A uncertain graph data stream is defined as UGDS= (V, E, P), PG=$\{g_1, g_2, \dots g_n\}$ ($g_i$=(V,$E_i$), n=$2^{|E|}$) is set of possible graphs that are instantiations of UGDS, P($g_i$) is probability with $g_i \in$ PG.so Expected Density of UGDS in a cluster is equal:

$$Expected\ \ Density \ = \frac{\sum_{i=1}^{m} P(g_i) \times 2 \times |E_i|}{(|v| \times (|v|-1))} \tag{2}$$

A simple formula to compute the *Expected Density* of UGDS in a cluster is :

$$Expected\ \ Density \ = \frac{\sum_{i=1}^{m} p_i \times 2}{(|v| \times (|v|-1))} \tag{3}$$

where $p_i$ represents the edge probability in the cluster, m is number of edges in a cluster and v is number of vertices in a cluster.

Using a describe in [6], *Clustering Similarity* inside the cluster and *Edge Similarity* with a cluster of UGDS are defined as follows :

Let C is a given micro cluster that contains G1,G2...Gn. Let H(C)=$\{$G1,G2,...Gn$\}$, n(c)= number of graphs in the micro cluster C then $\overline{H(C)}$ = edge frequency of H(C) divide n(c).

Let G is incoming graph then distance between the centroid graph $\overline{H(C)}$ and graph G is defined as follows:

$$L2Dist\ \ (G, \overline{H(C)}) = \sum_{i=1}^{m} (F(X_i, Y_i, G) - \frac{F(X_i, Y_i, H(C))}{n(c)})^2 \tag{4}$$

But the similarity function between the graph G and $\overline{H(C)}$ is defined as follows:

$$Dot\ \ (G, \overline{H(C)}) = \sum_{i=1}^{m} F(X_i, Y_i, G) \times \frac{F(X_i, Y_i, H(C))}{n(c)} \tag{5}$$

Next, we define *Clustering Similarity* in UGSC Algorithm with using (4),(5):

**Definition 3**. A uncertain graph data stream is defined as UGDS= (V, E, P), where P(E = $e_i$) = $p_i$, i=1, 2... m, P is a probability function expressing the relation between two entities or the edge probability in the cluster, then similarity function inside the cluster is defined as follow:

$$ClusteringSimilarity \ = \frac{1}{m} \sum_{j=1}^{m} p_j \times \frac{p_1 + p_2 + p_3 + \cdots + p_m}{m} \tag{6}$$

Where $p_i$ represents the edge probability ith in the cluster, it also was named *Clustering Similarity*.

For achieving the similarity function between new edge and a cluster we use from the average probabilities in a cluster to define *Edge Similarity* in Definition 4.

**Definition 4.** A uncertain graph data stream is defined as UGDS= (V, E, P), where P(E = $e_i$) = $p_i$, i=1, 2... m, P is a probability function expressing the relation between two entities or the edge probability in the cluster, then similarity function between a cluster and a new edge is defined as follows:

$$EdgeSimilarity \ = p' \times \frac{p_1 + p_2 + p_3 + \cdots + p_m}{m} \tag{7}$$

Where $p'$ is new edge in a cluster and $p_i$ is the edge probability ith in the cluster. it was named *Edge Similarity*.

## *3.6 Analysis of Running Time Complexity*

In this section, we study the time complexity of the algorithm. In the best case that no violation has happened for a cluster, each update is from the time order of O(1).

```
Algorithm Edge Insertion
Input: threshold, Uncertain Graph Stream UGS= (V, E, P), $\varepsilon$
if both vertices of the edge are new then
    if probability $>=$ threshold then
        insert them to vertex Table;
        insert the edge to cluster Table;
        increase the size of the cluster by 2;
    end if
else if only one vertex of the edge is new then
    if Expected Density with this new edge in cluster $>$ Expected Density without this new edge in
    cluster then
        insert the new vertex to vertex Table;
        insert the edge to the cluster of the old vertex;
        increase the size of the cluster by 1;
        if number of vertices in the cluster $>$ maximum cluster size then
            Edge List Copy = select recently edges of that cluster that max cluster size reach;
            call Deletion of an edge algorithm; (Algorithm Edge Deletion)
        end if
    else
        if (Clustering Similarity inside cluster $-$ Edge Similarity with cluster) $<\varepsilon$ then
            insert the new vertex to vertex Table;
            insert the edge to the cluster of the old vertex;
            increase the size of the cluster by 1;
            if number of vertices in the cluster $>$ maximum cluster size then
                Edge List Copy = select recently edges of that cluster that max cluster size reach;
                call Deletion of an edge algorithm; (Algorithm Edge Deletion)
            end if
        end if
    end if
else
    if (the two vertices are in the same cluster) and ( Clustering Similarity inside cluster without this
    edge $-$ Edge Similarity with cluster$<\varepsilon$ )
    then
        insert the edge to that cluster;
    else
        if (Clustering Similarity inside first cluster $-$ Edge Similarity with first cluster $<\varepsilon$ ) and
        ( Clustering Similarity inside second cluster $-$ Edge Similarity with second cluster$<\varepsilon$ ) then
            merge the smaller cluster to the bigger one;
            if number of vertices in the cluster $>$ maximum cluster size then
                Edge List Copy = select recently edges of that cluster that max cluster size reach;
                call Deletion of an edge algorithm; (Algorithm Edge Deletion)
            end if
        end if
    end if
end if
if window is full then
    Find the oldest edge of cluster from Cluster Table
    Edge List Copy = the edge list of that cluster;
    delete the first element from Edge List Copy
    call Deletion of an edge algorithm;( Algorithm Edge Deletion )
end if
```

**Fig. 4:** Edge Insertion in UGSC Algorithm

```
Algorithm Edge Deletion
delete the edges in cluster Table;
mark the corresponding vertices in vertex Table with Invalid ID;
for each edge in Edge List Copy do
    insert that edge using the insertion algorithm;
end for
```

**Fig. 5:** Edge Deletion in UGSC Algorithm

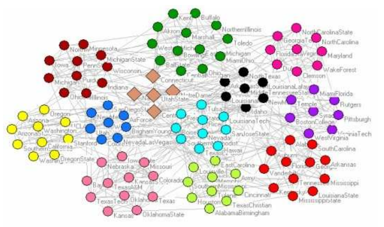
But for merging two clusters, the order time is O($m_1$+$m_2$) in which $m_1$ denotes the number of edges in the first cluster and $m_2$ denotes this number in the second one. In the worst case that violation has happened, if an edge is inserted in the previous cluster, then each update will be from the time order of O($m^2$) in which m denotes the number of edges in the cluster and if two clusters get merged in this case, the order time will be O($(m_1+m_2)^2$). The required space for this algorithm is equal to the memory needed for saving the table of vertices and table of clusters.

## 4 Experimental Results

We present a set of experiments to assess UGSC algorithm, including cluster evolution, the weighted cut size, stream Stableness and its throughput. The execution for each section was run on a Intel Core i5 and 4GB physical memory. All the algorithms were implemented in Matlab.

### *4.1 Data set*

First, we used Real Life Data include Karate Club [17] and College Football [17] as shown in Fig.6 The karate dataset contains friendships between 34 members of a karate club at a US university in the 1970s. There was a disagreement between the administrator and the instructor in the club, which resulted in two communities in this graph. The football dataset records games between Division IA colleges during regular season Fall 2000. There were 115 teams in 12 different conferences. UGSC algorithm detects communities correctly in Karate Club and College Football. second a real data sets in order to test UGSC algorithm. We implemented UGSC algorithm on two real data sets. The data sets are taken from Stanford University web site[14]. The real data sets used are DBLP and Youtube.


(a) Football

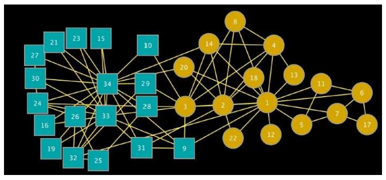

(a) Karate

**Fig. 6:** Real life data sets[17].

### *4.2 Evolution of Clusters*

To investigate the clustering evolution, we show three different snapshot and survey this times. In first snapshot, Fig.7 shows six clusters that every color represent different clusters. In this time we observe that the vertex labeled 13 is in the first cluster. After many updates vertex labeled 13 place in third cluster. This result shows in Fig.9.this is because of constraint violation of the first cluster happened in Fig.9.when the constraint violation occur the oldest edge and corresponding vertices delete from the cluster that vertex labeled 13 is one of them. Then Fig.9 shows vertex labeled 13 place in new cluster (the third cluster in Fig.9). In addition, we observe that Fig.8 show merge operation. The first cluster and sixth cluster in Fig.7 merge and make first cluster in Fig.8.

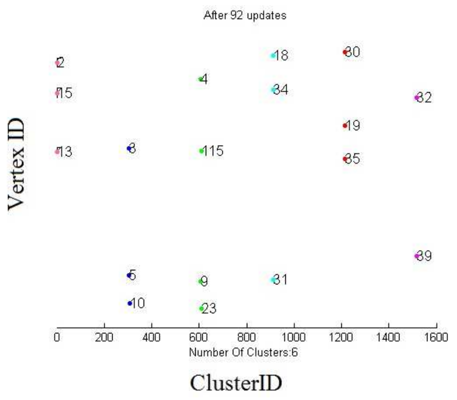


**Fig. 7:** Visualization of the Evolution in random graph in first snapshot.

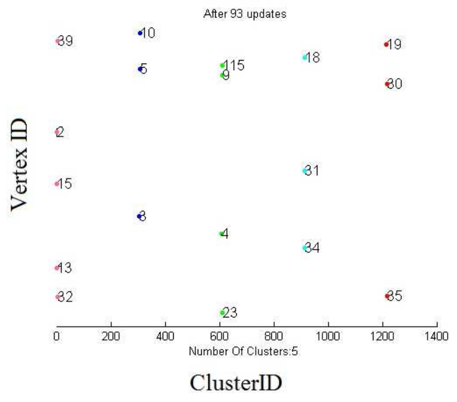


**Fig. 8:** Visualization of the Evolution in random graph in second snapshot.

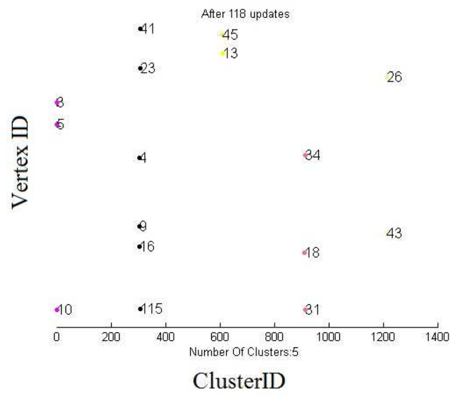


**Fig. 9:** Visualization of the Evolution in random graph in third snapshot.

## *4.3 Stream Stableness Experiments*

the stableness of the data sets is defined as follows[13]:

$$unstableness = Ave\left(\frac{numUnstableEdgesInWindow}{windowSize}\right) \tag{8}$$

$$stableness = -log(unstableness) \tag{9}$$

*Unstableness* is a metric based on a tumbling window because it's faster and gives the same final state. We could use a sliding window of the same size (e.g., 1K) which will give the same results but it will be much slower. Since for stableness experiments we don't measure the performance, we used a tumbling window so that we save our time while running the experiments to be able to run more experiments in a short time.Fig.10 shows stableness results in two data sets. By increasing the window size , the number of edges insertion in the clusters increase and the less changes occur in the number of clusters, so the graph stream is more stable.Fig.11 shows Youtube is more stable than the Dblp , because Youtube is dense graph in this experiments and the less changes occur in Youtube clusters.

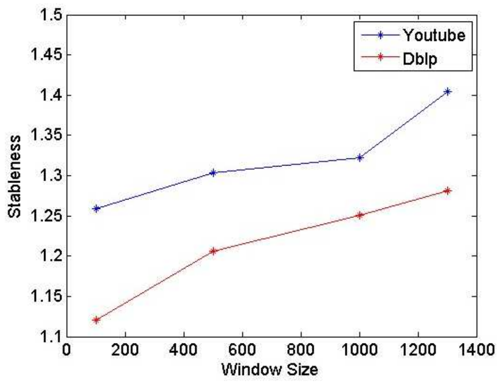


**Fig. 10:** Stream Stableness in different window size.

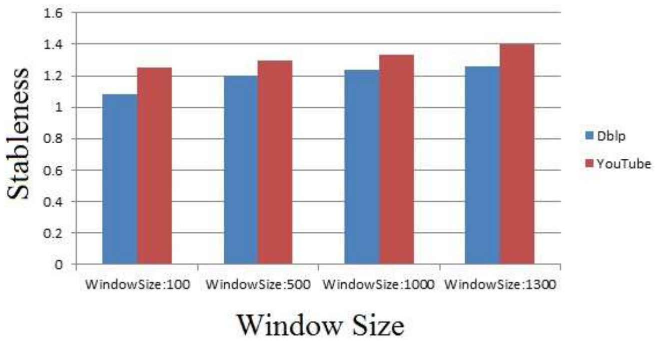


**Fig. 11:** Different stableness in different data sets: Youtube is more stable than Dblp

## *4.4 the weighted cut size Experiments*

Fig.12 shows Weighted cut size experiments with different maximum cluster sizes. By increasing the cluster size, the constraint violation decreases and this low constraint violation lead to the less edges deletion, so the weighted cut size in data sets decreases.in the best case There is enough memory and is not occurred constraint violation, then we capture the less weighted cut size. this weight cut is equal with weight cut in offline Agglomerative algorithm. offline Agglomerative algorithm was not designed for stream applications, but we can use it in a straight forward way. As edges are inserted or deleted, we keep a list of all edges in the graph. Whenever a query is issued, we run Agglomerative over the current set of edges and return the result. This is considered the implementation for graph clustering over streams. This result shows in Fig.13.In addition, Fig.13 shows the weighted cut when the threshold Changes.by decreasing the threshold, the weighted cut size decreases, so the number of edges in the clusters increase and lead to less inter-cluster edges.

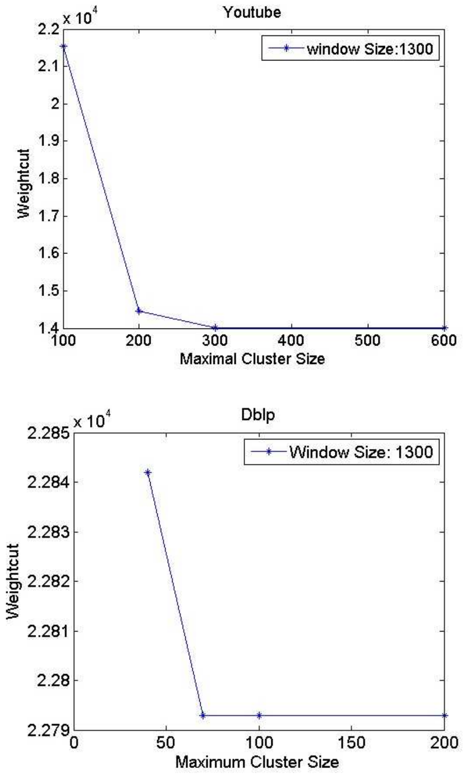


**Fig. 12:** Weighted Cut size with different maximum cluster sizes.

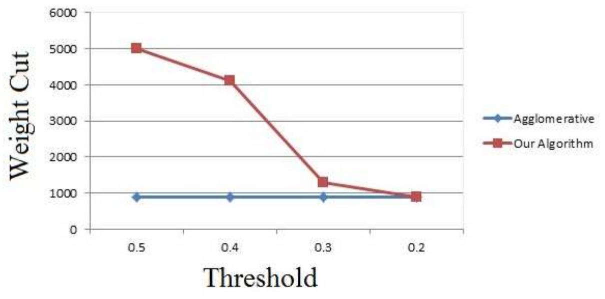


**Fig. 13:** Compare between Weighted Cut Sizes In Youtube dataset in different thresholds using Offline Agglomerative algorithm.

### 4.5 Performance Experiments

Fig.14 shows the throughput of the algorithm with different maximal cluster size. The bigger the clusters are, the lower the throughput is. It is because as the clusters size get bigger, the number of constraint violation reduces and when there exists the lower the edge deletion in the algorithm, the number of edges being deleted or reinserted decreases , therefore the throughput reduces. Also when the size of a cluster gets bigger, rate of throughput reduction becomes slower and this because of that the constraint violation reduces slowly by the size of clusters getting bigger and this makes the throughput more stable. The less dense a graph is, the less constraint violation it has and if a constraint violation takes place, the number of edges being deleted reduces so that the condition of clusters size is satisfied and also since the time taken for this work reduces, the throughput becomes higher. In our experiments, the throughput in Dblp is bigger than Youtube, so youtube is the most dense graph.

The bigger the Stream Stableness are, the lower the changes in clusters is, the less number of edges reinsert and the throughput is lower. the throughput in Dblp is bigger than Youtube, so youtube is more stable than Dblp. Figure 15 compares the throughput with offline Agglomerative algorithm. the throughput in UGSC algorithm is bigger than offline Agglomerative algorithm, because it is considered that there is the whole graph at the first and it takes a lot of the time.

### 4.6 Effect Of Parameter Threshold

Fig.16 shows the effect of sampling threshold on number of clusters. By increasing the threshold, number of edges in the clusters and total clusters reduce. Threshold is user defined in UGSC algorithm.

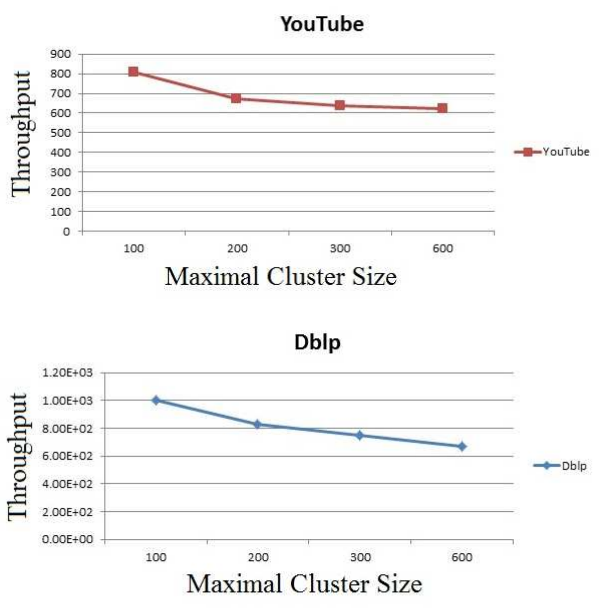


**Fig. 14:** Measurement of Throughput with different maximum cluster sizes.

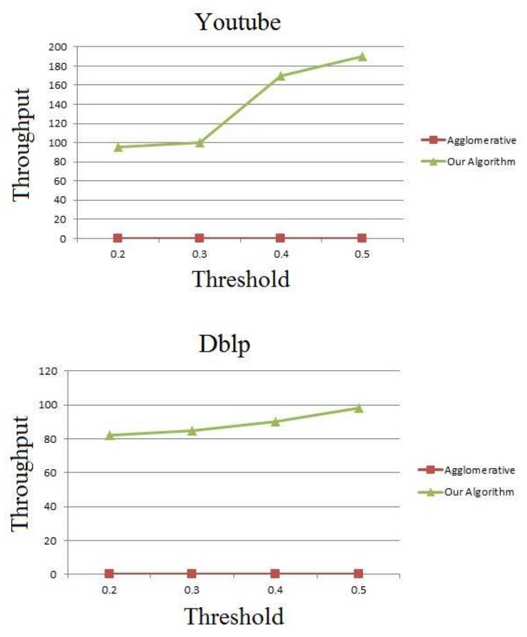


**Fig. 15:** Compare between throughput in different thresholds using offline Agglomerative algorithm.

## *4.7 Compare between EAC Algorithm And UGSC Algorithm In Certain Graph Data Stream*

Clustering vertices of a graph based on dynamic changes in edge connections is a powerful tool to understand social graphs, e.g., recognizing user communities. In scenarios where entity relationships change over time, a graph clustering algorithm must process a stream of updates. Each update can be the insertion or deletion of an edge or a vertex in the graph. Clustering of vertices in streaming graphs can be used to find user communities in real-time.

As a result, the real-world graphs have grown in size to millions or even billions of vertices and edges. Furthermore these graphs may grow or change with time rapidly. For example, Twitter with 200 million users as of 2011, generates over 200 million tweets and handles over 1.6 billion search queries per day. Here each tweet or search query can be thought of as an edge or a collection of edges in an appropriate graph[2]. While offline algorithms are not designed for stream applications so we compare UGSC algorithm with another online algorithm that is called EAC algorithm.

EAC is an evolution-aware clustering algorithm for processing of streaming certain graphs. [13]. When the edges of probability is 0 or 1, the graph is called certain. In this case we compare two algorithms. Fig.17 shows the number of clusters that is found in UGSC and EAC matches completely in two algorithms.

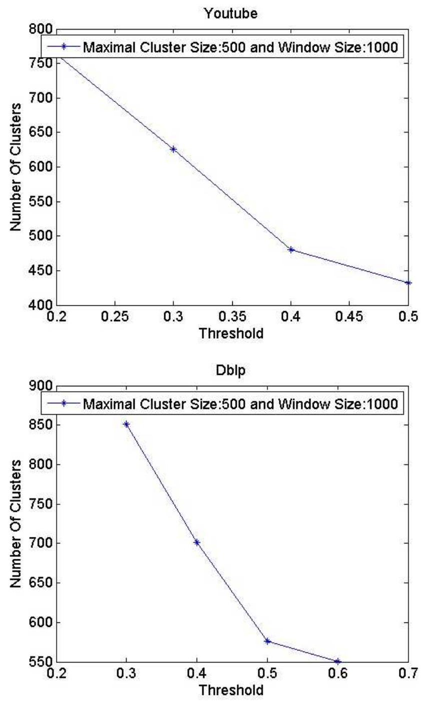


**Fig. 16:** These diagrams illustrate the effect of changing sampling threshold on Number Of Clusters.

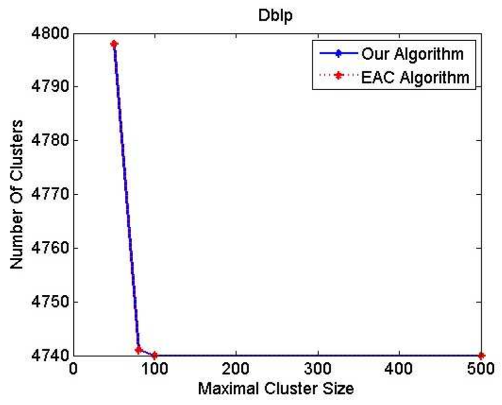


**Fig. 17:** Compare EAC Algorithm and UGSC Algorithm.

## 5 Conclusion and Future Works

Nowadays, according to the increasing spread of information which people deal with, taking advantage of methods such as data mining to extract hidden knowledge from data is necessary. Graph mining is a specific kind of data mining in which databases include data as graph. Each graph includes specific information of data. Presenting data as graph is usually used in social networks, bank transaction, interactions among proteins, drug discovery, etc. Denoting graphs does not always include uncertainty, therefore in this case the clustering algorithm that considers uncertainty is needed in order to get more reliable results. In the method proposed in this paper, graph data streams clustering with uncertainty is discussed. Obtained results are as follow:

- In the proposed method, it is shown that the less the stability of data stream is, the more the evolution in clusters happens.
- A new approach for clustering the uncertain graph data streams with concept drift and dynamics is presented.
- In the current algorithms presented for clustering the graph data, uncertainty is not discussed so much, and specially no such research has been done on graph data streams. For instance, in studying bank transactions like money laundering, the exact amount of exchanges is not mentioned, while uncertainty is the inseparable issue of graph data streams. In this research data uncertainty is modeled in a way that the proposed algorithm has reached the most reliable clusters with considering the uncertainty in data streams.
- The proposed algorithm is compared with an offline algorithm.
- Since an analysis has been done on the main data, the obtained results are reliable.

Future works are as follow:

- Proposing a method for clustering the distributed graph data streams with uncertainty. If the graph is huge such that it cannot be stored on the main memory, memory will be distributed on different systems. Since the table of vertices is much smaller than the table of clusters, so the vertices table is stored in one system and the clusters table is distributed among different systems. In this case the clusters are distributed by Hash Partitioning.
- Examining the proposed method in the case that uncertainty exists for both the edges and the vertices.
- Proposing a suitable dynamic method for extracting the repeating patterns in graph data streams is increasing such that patterns being repetitive or not repetitive are diagnosed based on the length of the pattern.
- Proposing a method for parallel processing in UGSC algorithm.

## References

[1] Angel, A. , Koudas, N. , Sarkas, N. and Srivastava, D. (2012),“Dense subgraph maintenance under streaming edge weight updates for real-time story indentication”, in 38th International Conference on Very Large Data in Proceedings of the VLDB Endowment in Istanbul, Turkey, 2012, Very Large Data Bases Endowment, USA , pp. 574-585.
[2] Eldawy, A. , Khandekar, R. , Wu, K.L. (2012),“Clustering Streaming Graphs”, in 32nd IEEE International Conference on Distributed Computing Systems in Proceedings of the IEEE International Conference in Macau, China, 2012, IEEE Computer Society, USA, pp. 466- 475.
[3] Bahmani,B. , Kumar ,R. and Vassilvitskii, S. (2012),“Densest subgraph in streaming and MapReduce”, in 38th International Conference on Very Large Data in Proceedings of the VLDB Endowment in Istanbul, Turkey, 2012, Very Large Data Bases Endowment, USA, pp. 454-465.
[4] Zhao, B. ,Wang, J., Wu F.X and Pan, Y. (2013), “Construction of Uncertain Protein-Protein Interaction Networks and Its Applications”, Bioinformatics Research and Applications, Springer Berlin Heidelberg ,USA, pp. 286-297.
[5] Aggarwal ,C. C., Zhao, Y. and Yu,P. S. (2010), “On clustering graph streams”, in International Conference on Data Mining in Proceedings of the SIAM in USA , 2010, SIAM,USA ,pp. 478-489.
[6] Aggarwal,C. C. , Zhao,Y. , Yu, P. S. (2010), “ A framework for clustering massive graph streams”, Published online 22 September 2010 in Wiley Online Library, in Statistical Analysis and Data Mining 2010, USA, pp. 399-416.
[7] Aggarwal,C. C. , Wang, H. (2010),“ Managing And Mining Graph Data , Springer, US.
[8] Aggarwal,C. C. (2014),“On Clustering Algorithms For Uncertain Data”, Springer, US.
[9] Gavin, A.C., Aloy ,P. (2006), “Proteome survey reveals modularity of the yeast cell machinery”, Heidelberg, Germany, pp.631-636 .

[10] Stanton ,I. and Kliot ,G. (2012), “Streaming graph partitioning for large distributed graphs ”, in International Conference on Knowledge Discovery and Data Mining in Proceedings of the 18th ACM SIGKDD in USA, 2012, ACM SIGMOD,USA, pp.1222-1230.
[11] Agarwal,M. K. , Ramamritham,K. and Bhide, M. (2012), “Real time discovery of dense clusters in highly dynamic graphs: identifying real world events in highly dynamic environments”. In Proceedings of the VLDB Endowment (PVLDB), in Istanbul, Turkey, 2012, Very Large Data Bases Endowment,USA,pp.980- 991.
[12] Gupta, M. , Aggarwal, C. C. , Han, J. and Sun,Y. (2011),“Evolutionary clustering and analysis of bibliographic networks”, in International Conference on Advances in Social Networks Analysis and Mining in ASONAM in USA, 2011, ASONAM, pp. 63-70.
[13] Yuan, M. ,Wu, K.L. , Lu,Y. , Jacques-Silva , G. (2013), “Efficient Processing of Streaming Graphs for Evolution-Aware Clustering”, in international conference on information and knowledge management in CIKM ’13 Proceedings of the 22nd ACM in USA, 2013,ACM, USA, pp. 319-328.
[14] Leskovec ,J. and Krevl, A. (2009),“Stanford Large Network Dataset Collection”, http://snap.stanford.edu/data , (accessed in 2014).
[15] Lin ,Y., Chi, Y. , Zhu, S. , Sundaram, H. and Tseng, B. (2008), “Facetnet: a framework for analyzing communities and their evolutions in dynamic networks”, 17th in international conference on World Wide Web in Proceedings of the WWW ’08 in USA, 2008,ACM, USA, pp. 685-694.
[16] Duan, L., Street , W. N., Lu, H., Liu, Y. (2014) ,“ Community Detection in Graphs through Correlation”, 20th ACM SIGKDD international conference on Knowledge discovery and data mining in KDD ’14 Proceedings in USA, 2014, ACM, USA, pp. 1376-1385.
[17] Huang, J., Deng , H., Sun, H., Sun, Y., Han,J., Liu,Y. (2010) ,“ SHRINK: A Structural Clustering Algorithm for Detecting Hierarchical Communities in Networks”, 19th ACM international conference on Information and knowledge management in CIKM ’10 Proceedings in USA, 2010, ACM, USA, pp. 219-228.
[18] Kollios, G., Potamias , M., Terzi, E. (2013) ,“Clustering Large Probabilistic Graphs”, in IEEE Transactions on Knowledge and Data Engineering in Proceedings of the IEEE International Conference in NJ, USA, 2013, IEEE Computer Society, USA, pp. 325-336.
[19] Liu, L., Aggarwal , C., Shen, Y. (2012) ,“ Reliable Clustering on Uncertain Graphs”, in 12th International Conference on Data Mining (ICDM ) in Proceedings of the IEEE International Conference in USA, 2012, IEEE Computer Society, USA, pp. 459-468.

**Zahra Varaminy Bahnemiry** received the Master degree in computer engineering at Science and Research Branch, Islamic Azad University in Iran. Her research interests are in the areas of Data Mining and Knowledge Discovery and Intelligent Systems.

**Mir Mohsen Pedram** has received his Ph.D in Electrical Engineering from Tarbiat Modarres University of Iran, in 2003. His major interests are Data Mining & Knowledge Discovery, Intelligent Systems, Machine Learning, Reinforcement Learning and has published more conference papers and journal papers. He is Assistant Professor of Computer Engineering Department, Kharazmi University of Iran. Email:pedram@khu.ac.ir

**Mitra Mirzarezaee** has received her Ph.D in computer engineering from Department of Computer engineering, Science and Research Branch, Islamic Azad University of Iran. She is Assistant Professor of Computer Engineering Department, Science and Research Branch, Islamic Azad University of Iran. Email: mirzarezaee@srbiau.ac.ir