Zhang *et al. J Big Data (2017) 4:11*
DOI 10.1186/s40537-017-0073-8

Journal of Big Data

RESEARCH Open Access

Outlier edge detection using random graph generation models and applications

Honglei Zhang[1*], Serkan Kiranyaz[2] and Moncef Gabbouj[1]

*Correspondence:
honglei.zhang@tut.fi
[1] Department of Signal Processing, Tampere University of Technology, Finland, Korkeakoulunkatu 1, FI-33101 Tampere, Finland
Full list of author information is available at the end of the article

Abstract

Outliers are samples that are generated by different mechanisms from other normal data samples. Graphs, in particular social network graphs, may contain nodes and edges that are made by scammers, malicious programs or mistakenly by normal users. Detecting outlier nodes and edges is important for data mining and graph analytics. However, previous research in the field has merely focused on detecting outlier nodes. In this article, we study the properties of edges and propose effective outlier edge detection algorithm. The proposed algorithms are inspired by community structures that are very common in social networks. We found that the graph structure around an edge holds critical information for determining the authenticity of the edge. We evaluated the proposed algorithms by injecting outlier edges into some real-world graph data. Experiment results show that the proposed algorithms can effectively detect outlier edges. In particular, the algorithm based on the Preferential Attachment Random Graph Generation model consistently gives good performance regardless of the test graph data. More important, by analyzing the authenticity of the edges in a graph, we are able to reveal underlying structure and properties of a graph. Thus, the proposed algorithms are not limited in the area of outlier edge detection. We demonstrate three different applications that benefit from the proposed algorithms: (1) a preprocessing tool that improves the performance of graph clustering algorithms; (2) an outlier node detection algorithm; and (3) a novel noisy data clustering algorithm. These applications show the great potential of the proposed outlier edge detection techniques. They also address the importance of analyzing the edges in graph mining—a topic that has been mostly neglected by researchers.

Keywords: Outlier detection, Graph mining, Outlier edge

Background

Graphs are an important data representation, which have been extensively used in many scientific fields such as data mining, bioinformatics, multimedia content retrieval and computer vision. For several hundred years, scientists have been enthusiastic about graph theory and its applications [1]. Since the revolution of the computer technologies and the Internet, graph data have become more and more important because many of the "big" data are naturally formed in a graph structure or can be transformed into graphs.

Outliers almost always happen in real-world graphs. Outliers in a graph can be outlier nodes or outlier edges. For example, outlier nodes in a social network graph may

include: scammers who steal users' personal information; fake accounts that manipulate the reputation management system; or spammers who send free and mostly false advertisements [2–4]. Researchers have been working on algorithms to detect these malicious outlier nodes in graphs [5–8]. Outlier edges are also common in graphs. They can be edges that are generated by outlier nodes, or unintentional links made by normal users or the system. Outlier edges are not only harmful but also greatly increase the system complexity and degrade the performance of graph mining algorithms. In this paper, we will show that the performance of the community detection algorithms can be greatly improved when a small amount of outlier edges are removed. Outlier edge detection can also help evaluate and monitor the behavior of end users and further identify the malicious entities. However, in contrast to the focus on the outlier node detection, there have been very few studies on outlier edge detection.

In this paper, we first propose an authentic score of an edge using the clustering property of social network graphs. The authentic score of an edge is determined by the difference of the actual and the expected number of edges that link the two groups of nodes that are around the investigating edge. We use random graph generation models to predict the number of edges between the two groups of nodes. The edges with low authentic scores, which are also called weak links in this paper, are likely to be outliers. We evaluated the outlier edge detection algorithm that is based on the authentic score using injected edges in real-world graph data.

Later, we show the great potentials of the outlier edge detection technique in the areas of graph mining and pattern recognition. We demonstrate three different applications that are based on the proposed algorithms: (1) a preprocessing tool for graph clustering algorithms; (2) an outlier node detection algorithm; (3) a novel noisy data clustering algorithm.

The rest of the paper is organized as follows: the prior art is reviewed in "Previous work"; the methodology to determine the authentic scores of edges is in "Methods"; evaluation of the proposed outlier edge detection algorithms are given in "Evaluation of the proposed algorithms"; various applications that use or benefit from outlier edge detection algorithms are presented in "Applications"; and finally, conclusions and future directions are included in "Conclusions".

Previous work

Outliers are data instances that are markedly different from the rest of the data [9]. Outliers are often located outside (mostly far way) from the normal data points when presented in an appropriate feature space. It is also commonly assumed that the number of outliers is much less than the number of normal data points.

Outlier detection in graph data includes outlier node detection and outlier edge detection. Noble and Cook studied substructures of graphs and used the Minimum Description Length technique to detect unusual patterns in a graph [6]. Xu et al. considered nodes that marginally connect to a structure (or community) as outliers [10]. They used a searching strategy to group the nodes that share many common neighbors into communities. The nodes that are not tightly connected to any community are classified as outliers. Gao et al. also studied the roles of the nodes in communities [11]. Nodes in a community tend to have similar attributes. Using the Hidden Markov Random Field

technique as a generative model, they were able to detect the nodes that are abnormal in their community. Akoglu et al. detected outlier nodes using the near-cliques and stars, heavy vicinities and dominant heavy links properties of the ego-network- the induced network formed by a focal node and its direct neighbors [12]. They observed that some pairs of the features of normal nodes follow a power law and defined an outlier score function that measures the deviation of a node from the normal patterns. Dai et al. detected outlier nodes in bipartite graphs using mutual agreements between nodes [7].

In contrast to proliferative research on outlier node detection, there have been very few studies on outlier edge detection in graphs. Liu et al. find outlier pairs in a complex network by evaluating the structural and semantic similarity of each pair of the connected nodes [13]. Chakrabarti detected outlier edges by partitioning nodes into groups using the Minimum Description Length technique [14]. Edges that link the nodes from different groups are considered as outliers. These edges are also called weak links or weak ties in literature [15]. Obviously this method has severe limitations. First, one shall not classify all weak links as outliers since they are part of the normal graph data. Second, many outlier edges do not happen between the groups. Finally, many graphs do not contain easily partitionable groups.

Detection of missing edges (or link prediction) is the opposite technique of outlier edge detection. These algorithms find missing edges between pairs of nodes in a graph. They are critical in recommendation systems, especially in e-commerce industry and social network service industry [16, 17]. Such algorithms evaluate similarities between each pair of nodes. A pair of nodes with high similarity score is likely to be connected by an edge. One may use the similarity scores to detect outlier edges. The edges whose two end nodes have a low similarity score are likely to be the outlier edges. However, in practice, these similarity scores do not give satisfactory performance if one uses them to detect outlier edges.

Methods

Notation

Let $G(V, E)$ denote a graph with a set of nodes V and a set of edges E. In this article, we consider undirected, unweighted graphs that do not contain self-loops. We use lower case a, b, c, etc., to represent nodes. Let $\overline{ab}$ denote the edge that connects nodes a and b. Because our graph G is undirected, $\overline{ab}$ and $\overline{ba}$ represent the same edge. Let N_a be the set of neighboring nodes of node a, such that $N_a = \{x | x \in V, \overline{xa} \in E\}$. Let $S_a = N_a \cup \{a\}$ (i.e. S_a contains node a and its neighboring nodes). Let k_a be the degree of node a, so that $k_a = |N_a|$. Let A be the adjacency matrix of graph G. Let $n = |V|$ be the number of nodes and $m = |E|$ be the number of edges of graph G.

Freeman defines the ego-network as the induced subgraph that contains a focal node and all of its neighboring nodes together with edges that link these nodes [18]. To study the properties of an edge, we define the edge-ego-network as follows:

Definition 1 An edge-ego-network is the induced subgraph that contains the two end nodes of an edge, all neighboring nodes of these two end nodes and all edges that link these nodes.

Let $G_{\overline{ab}} = G\left(V_{\overline{ab}}, E_{\overline{ab}}\right)$ denote the edge-ego-network of edge $\overline{ab}$, where $V_{\overline{ab}} = S_a \cup S_b$ and $E_{\overline{ab}} = \left\{\overline{xy} | x \in V_{\overline{ab}}, y \in V_{\overline{ab}} \text{ and } \overline{xy} \in E\right\}$.

Motivation

Graphs representing real-world data, in particular social network graphs, often exhibit the clustering property- nodes tend to form highly dense groups in a graph [19]. For example, if two people have many friends in common, they are likely to be friends too. Therefore, it is common for social network services to recommend new connections to a user using this clustering property [16]. As a consequence, social network graphs display an even stronger clustering property compared to other graphs. New connections to a node may be recommended from the set of neighboring nodes with the highest number of common neighbors to the given node. The common neighbors (CN) score of node a and node b is defined as

$$s_{CN} = |N_a \cap N_b|. \tag{1}$$

Common neighbors score is the basis of many node similarity scores that have been used to find missing edges [16]. Some common similarity indices are:

- Salton index or cosine similarity (Salton)

$$s_{Salton} = \frac{S_{CN}}{\sqrt{k_a k_b}} \tag{2}$$

- Jaccard index (Jaccard)

$$s_{Jaccard} = \frac{S_{CN}}{|N_a \cup N_b|} \tag{3}$$

- Hub promoted index (HPI)

$$s_{HPI} = \frac{S_{CN}}{\min(k_a, k_b)} \tag{4}$$

- Hub depressed index (HDI)

$$s_{HDI} = \frac{S_{CN}}{\max(k_a, k_b)} \tag{5}$$

Next we shall investigate how to detect outlier edges in a social network using the clustering property. According to this property, if two people are friends, they are likely to have many common friends or their friends are also friends of each other. If two people are linked by an edge, but do not share any common friends and neither do their friends know each other, we have good reason to suspect that the link between them is an outlier. So, when node a and node b are connected by edge $\overline{ab}$, there should be edges connect the nodes in set S_a and the nodes in set S_b. However, the number of connections should depend on the number of nodes in these two groups. Let us consider the different cases as shown in Fig. 1.

In these four cases, edge $\overline{ab}$ is likely to be a normal edge in case (d) because nodes a and b share common neighboring nodes c and d, and there are connections between

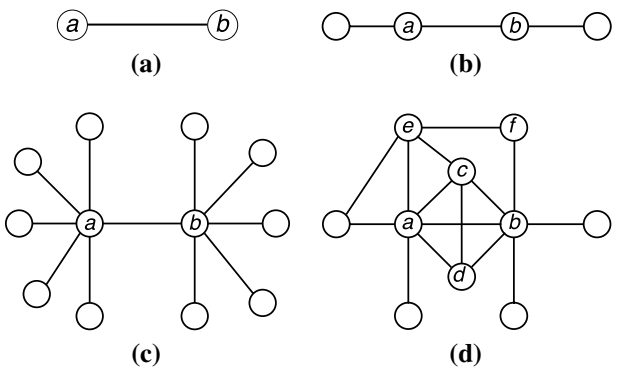

Fig. 1 Different cases of edge-ego-networks. (a) $k_a = k_b = 1, |N_a \cap N_b| = 0$ (b) $k_a = k_b = 2, |N_a \cap N_b| = 0$ (c) $k_a = k_b = 6, |N_a \cap N_b| = 0$ (d) $k_a = k_b = 6, |N_a \cap N_b| = 2$

neighboring nodes of a and those of b. In the case of (a), (b) and (c), $|N_a \cap N_b| = 0$, which implies that nodes a and b do not share any common neighboring nodes. However edge $\overline{ab}$ in case (c) is more likely to be an outlier edge because nodes a and b have each many neighboring nodes but there is no connection between any two of these neighboring nodes. In case (a) and (b) we do not have enough information to judge whether edge $\overline{ab}$ is an outlier edge or not. If we apply the node similarity scores to detect outlier edges, we find that $S_{CN} = 0$ for cases (a), (b) and (c). Thus, the node similarity scores defined by Eqs. (1), (2), (3), (4) and (5) all equal to 0. For this reason, these node similarity scores cannot effectively detect outlier edges.

In case (c), edge $\overline{ab}$ is likely to be an outlier edge because the expected number of edges between node a together with its neighboring nodes and node b together with its neighboring nodes is high, whereas the actual number of edges is low. So, according to the clustering property, we propose the following definition for the authentic score of an edge:

Definition 2 The authentic score of an edge is defined as the difference between the number of actual edges and the expected value of the number of edges that link the two sets of neighboring nodes of the two end nodes of the given edge. That is:

$$s_{\overline{ab}} = m_{\overline{ab}} - e_{\overline{ab}}, \tag{6}$$

where $m_{\overline{ab}}$ is the actual number of edges that links the two sets of nodes- one set is node a together with its neighboring nodes and the other set is node b together with its neighboring nodes, and $e_{\overline{ab}}$ is the expected number of edges that link the aforementioned two sets of nodes.

We can rank the edges by their authentic scores defined in Eq. (6). The edges with low scores are more likely to be outlier edges in a graph.

Let $\alpha(S, T) = \left|\overline{ab} | a \in S, b \in T \text{ and } \overline{ab} \in E\right|$ denote the number of edges that links the nodes in sets S and T. We suppose the graph G is generated by a random graph generation model. Let $\epsilon(S, T)$ denote the expected value of the number of edges that links the nodes in sets S and T by the generation model. "Expected number of edges between two

sets of nodes" describes two generation models and the functions of calculating $\epsilon(S,T)$. Obviously $\alpha(S,T)$ and $\epsilon(S,T)$ are symmetric functions. That is:

Theorem 1 $\alpha(S,T) = \alpha(T,S)$ *and* $\epsilon(S,T) = \epsilon(T,S)$.

Let $P_{a,b}$ and $R_{a,b}$ be the two sets of nodes that are related to end nodes a and b. Node set $R_{a,b}$ depends on set $P_{a,b}$. The actual number of edges and the expected number of edges of the sets of nodes related to the two end nodes may vary when we switch the end nodes a and b. We use the following equations to calculate $m_{\overline{ab}}$ and $e_{\overline{ab}}$:

$$m_{\overline{ab}} = \frac{1}{2}\left(\alpha\left(P_{a,b}, R_{a,b}\right) + \alpha\left(P_{b,a}, R_{b,a}\right)\right); \tag{7}$$

$$e_{\overline{ab}} = \frac{1}{2}\left(\epsilon\left(P_{a,b}, R_{a,b}\right) + \epsilon\left(P_{b,a}, R_{b,a}\right)\right). \tag{8}$$

Schemes of node neighborhood sets

For a ego-network, Coscia and Rossetti showed the importance of removing the focal node and all edges that link to it when studying the properties of ego-networks [20]. It is more complicate to study the properties of an edge-ego-network since there are two ending nodes and two sets of neighboring nodes involved. Considering the common nodes of the neighboring nodes and the end nodes of the edge being investigated, we now define four schemes that capture different configurations of these two sets.

Let $S_{a\setminus b} = S_a \setminus \{b\}$ be the set of nodes that contains node a and its neighboring nodes except node b. Let $N_{a\setminus b} = N_a \setminus \{b\}$ be the set of nodes that contains the neighboring nodes of a except node b. Obviously $S_{a\setminus b} = N_{a\setminus b} \cup \{a\}$. Fig. 2 shows the edge-ego-network $G_{\overline{ab}}$ and the two sets of nodes $S_{a\setminus b}$ and $S_{b\setminus a}$ corresponding to case (d) in Fig. 1.

We first define two sets of nodes that are related to node a and its neighboring nodes: $N_{a\setminus b}$ and $S_{a\setminus b}$. Next, we define two sets of nodes that are related to node b and its neighboring nodes with regard to the sets of nodes $N_{a\setminus b}$ and $S_{a\setminus b}$: $S_{b\setminus a}\setminus S_{a\setminus b}$ and $S_{b\setminus a}$. In Fig. 2, $N_{a\setminus b} = \{c,d,e,g,h\}$, $S_{a\setminus b} = \{a,c,d,e,g,h\}$, $S_{b\setminus a}\setminus S_{a\setminus b} = \{b,f,i,j\}$ and

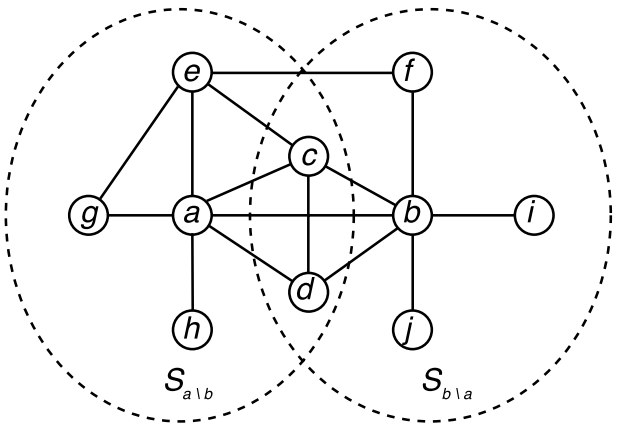

Fig. 2 The sets of the nodes of the edge-ego-network $G_{\overline{ab}}$ in the case (d) of Fig. 1. $S_{a\setminus b}$ contains node a and its neighboring nodes except node b; $S_{b\setminus a}$ contains node b and its neighboring nodes except node a

$S_{b\backslash a} = \{b, c, d, f, i, j\}$. In the case of a social network graph, $N_{a\backslash b}$ would consist of friends of user (node) a except b; $S_{a\backslash b}$ consists of a and friends of a except b; $S_{b\backslash a} \backslash S_{a\backslash b}$ consists of b and friends of b except a and those who are friends of a; $S_{b\backslash a}$ consists of b and friends of b except a.

Based on the set pairs of nodes a and b, we define the following four schemes and their meanings in the case of a social network graph. We use superscript (1), (2), (3) and (4) to indicate the four schemes respectively.

- Scheme 1 : $P^{(1)}_{a,b} = N_{a\backslash b}$ and $R^{(1)}_{a,b} = S_{b\backslash a} \backslash S_{a\backslash b}$
  How many of a's friends know b and his friends outside of the relationship with a?

- Scheme 2 : $P^{(2)}_{a,b} = N_{a\backslash b}$ and $R^{(2)}_{a,b} = S_{b\backslash a}$
  How many of a's friends know b and his friends?

- Scheme 3 : $P^{(3)}_{a,b} = S_{a\backslash b}$ and $R^{(3)}_{a,b} = S_{b\backslash a} \backslash S_{a\backslash b}$
  How many of a and his friends know b and his friends outside of the relationship with a?

- Scheme 4 : $P^{(4)}_{a,b} = S_{a\backslash b}$ and $R^{(4)}_{a,b} = S_{b\backslash a}$
  How many of a and his friends know b and his friends?

For the edge-ego-network $G_{\overline{ab}}$ shown in Fig. 2, scheme 1 examines edges $\overline{ef}$, $\overline{cb}$ and $\overline{db}$; scheme 2 examines edges $\overline{ef}$, $\overline{ec}$, $\overline{cb}$, $\overline{cd}$, $\overline{dc}$ and $\overline{db}$; scheme 3 examines edges $\overline{ab}$, $\overline{ef}$, $\overline{cb}$ and $\overline{db}$; scheme 4 examines edges $\overline{ab}$, $\overline{ac}$, $\overline{ad}$, $\overline{ef}$, $\overline{ec}$, $\overline{cb}$, $\overline{db}$, $\overline{dc}$ and $\overline{cd}$.

Next we study the symmetric property of these four schemes.

Theorem 2 $\alpha\left(P^{(2)}_{a,b}, R^{(2)}_{a,b}\right) = \alpha\left(P^{(2)}_{b,a}, R^{(2)}_{b,a}\right)$ *and* $\alpha\left(P^{(4)}_{a,b}, R^{(4)}_{a,b}\right) = \alpha\left(P^{(4)}_{b,a}, R^{(4)}_{b,a}\right)$

The proof of this theorem is given in Appendix. Theorem 2 shows that the number of edges that link the nodes from the two groups defined in scheme 2 and scheme 4 are symmetric. That is the values remains the same if the two end nodes are switched. We can use $m^{(2)}_{\overline{ab}} = \alpha\left(P^{(2)}_{a,b}, R^{(2)}_{a,b}\right)$ and $m^{(4)}_{\overline{ab}} = \alpha\left(P^{(4)}_{a,b}, R^{(4)}_{a,b}\right)$ instead of Eq. 7.

Theorem 3 $\epsilon\left(P^{(4)}_{a,b}, R^{(4)}_{a,b}\right) = \epsilon\left(P^{(4)}_{b,a}, R^{(4)}_{b,a}\right)$

This theorem can be directly derived from $P^{(4)}_{a,b} = R^{(4)}_{b,a}$, $R^{(4)}_{a,b} = P^{(4)}_{b,a}$ and Theorem 1. So $e_{\overline{ab}} = \epsilon\left(P^{(4)}_{a,b}, R^{(4)}_{a,b}\right)$. Note scheme 4 is symmetric in calculating both of the actual and expected number of edges of the two groups.

Expected number of edges between two sets of nodes

With the four schemes described above, we get the number of edges that connect nodes from the two sets using Eq. 7. To calculate the authentic score of an edge by Eq. (6), we should find the expected number of edges between these two sets of nodes. Next we will use random graph generation models to determine the expected number of edges between these two sets of nodes.

Erdős- Rényi random graph generation model

The Erdős-Rnyi model, often referred as $G(n, m)$ model, is a basic random graph generation model [21]. It generates a graph of n nodes and m edges by randomly connecting two nodes by an edge and repeat this procedure until the graph contains m edges.

Suppose we have n nodes in an urn and predefined two sets of nodes S and T. We randomly pick two nodes from the urn. Note, the intersection of sets S and T may not be empty. The probability of picking the first node from set $S \backslash T$ is $\frac{|S|-|S\cap T|}{n}$ and the probability of picking the first node from set $S \cap T$ is $\frac{|S\cap T|}{n}$. If the first node is from set S, the probability of picking the second node from set T is $\frac{|S|-|S\cap T|}{n}\frac{|T|}{n-1} + \frac{|S\cap T|}{n}\frac{|T|-1}{n-1}$. Since the graph is undirected, we may also pick up a node from set T first and then pick up the second node from set S. So, the probability that we generate an edge that connects a node set S and a node from set T by randomly picking is:

$$p(S, T) = (|S||T| - |S \cap T|)\frac{2}{n(n-1)}. \tag{9}$$

We repeat this procedure m times to generate a graph, where m is the number of edges in graph G. The expected number of edges that connect the nodes in set S and the nodes in set T is:

$$\epsilon(S, T) = (|S||T| - |S \cap T|)\frac{2m}{n(n-1)}. \tag{10}$$

Note, here we ignore the duplicate edges during this procedure. This has little impact on the final results for real-world graphs where $m \ll n(n-1)$. In Eq. (10), let

$$d_G = \frac{2m}{n(n-1)}, \tag{11}$$

where d_G is the density (or fill) of graph G.

Next we will find the expected number of edges under the four schemes defined in "Schemes of node neighborhood sets". Since edge $\overline{ab}$ is already fixed, we should repeat the random procedure $m-1$ times. For real-world graphs where $m \gg 1$, we can safely approximate $m-1$ by m.

Now we can apply Eq. (10) under the four schemes. Let k_a and k_b be the degrees of nodes a and b. Let $k_{ab} = |N_a \cap N_b|$ be the number of common neighboring nodes of nodes a and b. The expected number of edges for each scheme is:

- Scheme 1:

$$e_{\overline{ab}}^{(1)} = \left(k_a k_b - \frac{1}{2}(k_a + k_b)(1 + k_{ab}) + k_{ab}\right)d_G \tag{12}$$

- Scheme 2:

$$e_{\overline{ab}}^{(2)} = \left(k_a k_b - \frac{1}{2}(k_a + k_b) - k_{ab}\right)d_G \tag{13}$$

- Scheme 3:

$$e_{\overline{ab}}^{(3)} = \left(k_a k_b - \frac{1}{2}(k_a + k_b)k_{ab}\right)d_G \tag{14}$$

- Scheme 4:

$$e_{\overline{ab}}^{(4)} = (k_a k_b - k_{ab}) d_G \tag{15}$$

Preferential attachment random graph generation model

The Erdős- Rnyi model generates graphs that are lacking some important properties of real-world data, in particular the power law of the degree distribution [1]. Next we introduce a random graph generation model using a preferential attachment mechanism that generates a random graph in which degrees of each node are known. Our preferential attachment random graph generation model (PA model) is closely related to the modularity measurement that evaluates the community structure in a graph. Newman defines the modularity value as the difference of the actual number of edges and the expected number of edges of two communities [22]. The way of calculating the expected number of edges between two communities follows preferential attachment mechanism instead of using the Erdős- Rényi model. In the Erdős- Rényi model, each node is picked with the same probability. However, by the preferential attachment mechanism, the nodes with high degrees are picked with high probabilities. Thus an edge is more likely to link nodes with a high degree.

We can apply the preferential attachment strategy to generate a random graph with n nodes, m edges and each node has a predefined degree value. We first break each edge into two ends and put all the $2m$ ends into an urn. A node with degree k will have k entities in the urn. At each round, we randomly pick two ends (one at a time with substitution) from the urn, link them with an edge and put them back into the urn. We repeat this procedure m times. We call this procedure Preferential Attachment Random Graph Generation model, or PA model in short. Note, we may generate duplicate edges or even self-loops with this procedure. Thus the expected number of edges estimated by this model is higher than a model that does not generate duplication edges and self-loops. This defect can be ignored when k_a and k_b are small. Later we will show a method that can compensate this bias, especially when k_a and k_b are large.

If we have two nodes a and b, the probability that an edge is formed in each round is:

$$p_{\overline{ab}} = \frac{k_a k_b}{2m^2}. \tag{16}$$

Then the expected number of edges that link the nodes a and b after m iterations is:

$$e_{\overline{ab}} = \frac{k_a k_b}{2m}. \tag{17}$$

If we have two sets of nodes S and T, the expected number of edges that link the nodes in set S and the nodes in set T is:

$$\epsilon(S, T) = \sum_{a \in S} \sum_{b \in T} e_{\overline{ab}} = \frac{1}{2m} \sum_{a \in S} \sum_{b \in T} k_a k_b. \tag{18}$$

Applying Eq. (18) to the four schemes defined in "Schemes of node neighborhood sets", we get the expected number of edges for each scheme is

- Scheme 1:

$$e_{\overline{ab}}^{(1)} = \frac{1}{4m}\left(\sum_{i\in P_{a,b}^{(1)}}\sum_{j\in R_{a,b}^{(1)}} k_i k_j + \sum_{i\in P_{b,a}^{(1)}}\sum_{j\in R_{b,a}^{(1)}} k_i k_j\right) \tag{19}$$

- Scheme 2:

$$e_{\overline{ab}}^{(2)} = \frac{1}{4m}\left(\sum_{i\in P_{a,b}^{(2)}}\sum_{j\in R_{a,b}^{(2)}} k_i k_j + \sum_{i\in P_{b,a}^{(2)}}\sum_{j\in R_{b,a}^{(2)}} k_i k_j\right) \tag{20}$$

- Scheme 3:

$$e_{\overline{ab}}^{(3)} = \frac{1}{4m}\left(\sum_{i\in P_{a,b}^{(3)}}\sum_{j\in R_{a,b}^{(3)}} k_i k_j + \sum_{i\in P_{b,a}^{(3)}}\sum_{j\in R_{b,a}^{(3)}} k_i k_j\right) \tag{21}$$

- Scheme 4:

$$e_{\overline{ab}}^{(4)} = \frac{1}{4m}\left(\sum_{i\in P_{a,b}^{(4)}}\sum_{j\in R_{a,b}^{(4)}} k_i k_j + \sum_{i\in P_{b,a}^{(4)}}\sum_{j\in R_{b,a}^{(4)}} k_i k_j\right) \tag{22}$$

Authentic score using the PA model

Authentic score compensation

We may apply Eqs. (19), (20), (21) or (22) to Eq. (6) to calculate the authentic score of an edge. As mentioned in "Preferential attachment random graph generation model", the PA model generates graphs with duplicate edges and self-loops. Thus the estimated expected number of edges that link two sets of nodes are higher than an accurate model. The gap is even more significant when the number of edges is large. To compensate for this bias, we refine the authentic score function for the PA model as

$$s_{\overline{ab}} = m_{\overline{ab}}^{\gamma} - e_{\overline{ab}}, \tag{23}$$

where $\gamma > 1$. The power function of the first term increases the value, especially when $m_{\overline{ab}}$ is large. This eventually compensates the bias introduced in the second term. In practice, we normally choose $\gamma = 2$.

Matrix of degree products

To get $e_{\overline{ab}}$ using Eqs. (19), (20), (21) or (22), we should find the sum of $k_a k_b$ for every pair of nodes in the corresponding edge-ego-network. We can store the values of $k_a k_b$ for every pair of nodes to prevent unnecessary multiplication operations and thus reduce the processing time. However, storing this information would require a storage space in the order of n^2, which is not applicable when n is large. We observe that we do not need

to calculate the product of the degrees for every pair of nodes in graph G. What we need is the pair of nodes that appear together in every edge-ego-network.

The distance of two nodes in a graph is defined as the length of the shortest path between them. It is easy to see that the maximum distance of two nodes in an edge-ego-network is 3. Next, we use the property of the adjacency matrix to find the pairs of nodes that appear together in edge-ego-networks.

Let d_{ij} be the distance of node i and node j. Let $B(k) = A^k$, where A is the adjacency matrix of graph G and k is a natural number. Let $B_{ij}(k)$ be the element of the matrix $B(k)$. Then $B_{ij}(k)$ is the number of walks with length k between node i and node j. If $B_{ij}(k) = 0$, there is no walk with length k between nodes i and j.

Proposition 3.1 *If* $d_{ij} = k$, $B_{ij}(k) \neq 0$

Proof If $d_{ij} = k$, there exists at least one path with length k from node i to node j. Since a path of a graph is a walk between two nodes without repeating nodes, there exists at least one walk with length k between the node i and the node j. So $B_{ij}(k) \neq 0$.

Theorem 4 *Let* $K(k) = B(1) + B(2) + \cdots + B(k)$. If $d_{ij} \leq k$, $K_{ij}(k) \neq 0$

Proof Let $d_{ij} = l$, where $l \leq k$. From Proposition 3.1, $B_{ij}(l) \neq 0$. Since $B(k)$ is a nonnegative matrix where $B_{ij}(k) \geq 0$, we have $K_{ij}(k) = B_{ij}(1) + \cdots + B_{ij}(l) + \cdots + B_{ij}(k) \neq 0$.

According to Theorem 4, to find the pairs of nodes with a distance of 3 or less, we need to find the nonzero elements in matrix $K(3)$. Let I be the indicator matrix whose elements indicate whether the distance between a pair of nodes is equal to or less than 3. Such that:

$$I_{ij} = \begin{cases} 1 & if\ K_{ij}(3) \neq 0 \\ 0 & if\ K_{ij}(3) = 0 \end{cases}. \tag{24}$$

Let matrix D denote the degree matrix whose diagonal elements are the degree of each node, that is:

$$D_{ij} = \begin{cases} k_i & if\ i = j \\ 0 & otherwise \end{cases}. \tag{25}$$

Let

$$E = \frac{1}{2m}\Big((DI) \circ (DI)^T\Big), \tag{26}$$

where $\circ$ denotes the Hadamard product of two matrices. The value of the nonzero elements in matrix E is the expected number of edges between the two nodes under the PA model. Using matrix E, we can easily calculate the authentic score for each scheme. For example the authentic score of the edge $\overline{ab}$ using scheme 1 and the score function defined by Eq. (6) is:

$$s_{ab}^{(1)} = \frac{1}{2}\left(\sum_{i\in P_{a,b}^{(1)}}\sum_{j\in R_{a,b}^{(1)}}\left(A_{ij}-E_{ij}\right) + \sum_{i\in P_{b,a}^{(1)}}\sum_{j\in R_{b,a}^{(1)}}\left(A_{ij}-E_{ij}\right)\right). \quad (27)$$

Evaluation of the proposed algorithms

In this section we evaluate the performance of the proposed outlier edge detection algorithms. Due to the availability of the datasets with identified outlier edges, we generate test data by injecting random edges to real-world graphs. This experimental setup is effective to evaluate algorithms that detect outliers, since the injected edges are random thus do not follow the actual principle that generated the real-world graph. We also evaluate the proposed outlier detection algorithms by measuring the change of some important graph properties when outlier edges are removed. In next section, we will show that the proposed algorithms are not only effective in simulated data but also powerful in solving real-world problems in many areas.

We first inject edges to a real-world graph data by randomly picking two nodes from the graph and linking them with an edge, if they are not linked. The injected edges are formed randomly, and thus they do not follow any underlying rule that generated the real-world graph. An outlier edge detection algorithm returns the authentic score of each edge. Given a threshold value, the edges with lower scores are classified as outliers.

With multiple algorithms, we vary the threshold value and record the true positive rates and the false positive rates of each algorithm. We use the receiver operating characteristic (ROC) curve—a plot of true positive rates against false positive rates at various threshold values—to subjectively compare the performance of different algorithms. We also calculate the area under the ROC curve (AUC) value to quantitatively evaluate the competing algorithms.

Comparison of different combinations of the proposed algorithm

The proposed algorithm involves two random graph generation models and four schemes. Two authentic score functions are proposed for the PA Model. With the first experiment, we study the performance of different combinations using real-world graph data.

We take the Brightkite graph data as the test graph [23]. Brightkite is a social network service in which users share their location information with their friends. The Brightkite graph contains 58, 228 nodes and 214, 708 edges. The data was received from the KONECT graph data collection [24].

We injected 1000 random "false" edges to the graph data. If an algorithm yields the same authentic scores to multiple edges, we randomly order these edges. We compare the detection results of the algorithms using the Erdős- Rényi (ER) model and the PA model with the combination of the four schemes explained in "Schemes of node neighborhood sets" and the two score functions defined in Eqs. (6) and (23). Table 1 shows the AUC values of the ROC curves of all combinations. Italic font indicates the best score among all of them.

From the experimental results, we see that the performance of the PA model with score function defined by Eq. (23) is clearly better than that of the score function defined by Eq. (6). The term m^{γ} in Eq. (23) increases the value even more when m is large. After

Table 1 AUC values of the ROC curves using Brightkite graph Data

	ER model		PA model	
	Eq. (6)	**Eq. (23)**	**Eq. (6)**	**Eq. (23)**
Scheme 1	0.885	0.885	0.880	0.904
Scheme 2	0.885	0.885	0.882	*0.905*
Scheme 3	0.878	0.878	0.873	0.902
Scheme 4	0.879	0.879	0.878	0.903

Italic indicates the best score of each experiment

the bias of the PA model is corrected, the performance of the outlier edge detection algorithm is greatly improved. The choice of the score function defined by Eqs. 6 and 23 has little impact to the ER model based algorithms.

The results also show that the combination of the PA model and the score function defined by Eq. (23) is superior than other combinations by a significant margin. Scheme 2 gives better performance than the other schemes, especially for ER Model based algorithms. In the rest of this paper, we use scheme 2 for the ER Model based algorithm. With the combination of the PA Model and the score function defined by Eq. 23, the difference between each scheme is insignificant. Because of the symmetric property of scheme 4, we use it for the PA model with the score function defined by Eq. 23.

Comparison of outlier edge detection algorithms

In this section we perform comparative evaluation of the proposed outlier edge detection algorithms against other algorithms. All test graphs originate from the KONECT graph data collection. Table 2 shows some parameters of the test graph data. The density of a graph is defined in Eq. (11). GCC, which stands for the global clustering coefficient, is a measure of clustering property of a graph. It is the ratio of the number of closed triangles and the number of connected triplet nodes. The higher GCC value is, the stronger clustering property a graph has.

We compared the performance of the two proposed algorithms [ER model combined with scheme 2 and the score function defined by Eq. (6) and PA model combined with scheme 4 and the score function defined by Eq. (23)] with three other algorithms that use node similarity scores for missing edge detection. We use the Jaccard Index and Hub Promoted Index (HPI) as defined in Eqs. (3) and (4). We also use the preferential

Table 2 Test graph data for comparing outlier edge detection Algorithms

	Nodes	**Edges**	**Density**	**GCC (%)**	**Reference**
Advogato	6.5 k	51 k	1.2×10^{-3}	9.2	[25]
Twitter-icwsm	465 k	835 k	3.9×10^{-6}	0.06	[26]
Brightkite	58 k	214 k	1.3×10^{-4}	11	[23]
Facebook-wosn	63 k	817 k	4.0×10^{-4}	14.8	[27]
Ca-cit-HepPh	28 k	4.6 m	8.0×10^{-3}	28	[28]
Youtube-friend	1.1 m	3.0 m	4.6×10^{-6}	0.6	[29]
Web-Google	875 k	5.1 m	6.7×10^{-6}	5.5	[30]

attachment index (PAI) that is another missing edge detection metric that works for outlier edge detection. The PAI for edge $\overline{ab}$ is defined as

$$s_{PAI} = k_a k_b. \tag{28}$$

Figure 3 shows the ROC curves of different algorithms on the Brightkite graph data. For reference, the figure also shows an algorithm that randomly orders the edges by giving random scores to each edge.

As Fig. 3 shows, the ROC curve of the algorithm that gives random scores is roughly a straight line from the origin to the top right corner. This line indicates that the algorithm cannot distinguish between an outlier edge and a normal edge, which is expected. The ROC curve of an algorithm that can detect outlier edges should be a curve above this straight line, as all algorithms used in this experiment. As mentioned in "Motivation", the Jaccard Index and HPI both use the number of common neighbors. Thus their scores are all 0 for edges that connect two end nodes that do not share any common neighbors. In real-world graphs, a large amount of edges have a Jaccard Index or HPI value 0, especially for graphs that contain many low degree nodes.

The PAI value is the product of the degrees of the two end nodes of an edge. Sorting edges with their PAI values just puts the edges with low degree end nodes to the front. The figure shows that the PAI value can detect outlier edges with fairly good performance. This indicates that most of the injected edges connecting the nodes with low degrees. Considering most of the nodes in a real-world graph are low degree nodes, this is an expected behavior.

Figure 3 indicates that the proposed outlier edge detection algorithms are clearly superior to the competing algorithms. The algorithm based on the PA model performs better than the one based on the ER model.

Table 3 shows the AUC values of the ROC curves on all test graph data. Italic font shows the best AUC values for each test graph.

The comparison results show that the PA model algorithm gives consistently good performance regardless of the test graph data. The experiment also shows the correlation

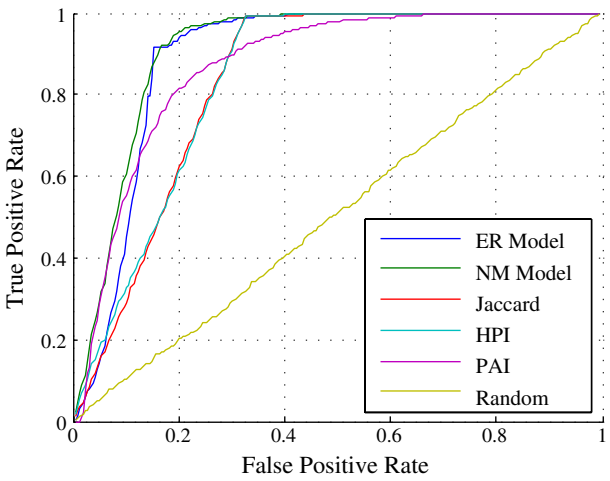

Fig. 3 ROC curve of different algorithms on the Brightkite graph data. The curves of the proposed methods are clearly above other competing methods

Table 3 AUC values of the ROC curves on different graph data

	ER	PA	Jaccard	HPI	PAI
Advogato	0.887	*0.893*	0.858	0.859	0.877
Twitter-icwsm	0.531	0.942	0.527	0.530	*0.997*
Brightkite	0.885	*0.905*	0.833	0.827	0.873
Facebook-wosn	0.968	*0.970*	0.947	0.946	0.878
Ca-cit-HepPh	0.970	0.967	*0.993*	0.991	0.888
Youtube-friend	0.770	0.842	0.731	0.738	*0.898*
Web-Google	0.985	*0.992*	0.944	0.945	0.859

Italic indicates the best score of each experiment

between the performance of the algorithms that are based on the random graph generation model and the GCC value of the test graph. For example, the ER model and PA model algorithms works better on Facebook-Wosn and Brightkite graph data, which have high GCC values as shown in Table 2. Performance of the ER model algorithm degrades considerably on graphs with a very low GCC value, such as the twitter-icwsm graph. This result agrees with the fact that both the ER model and the PA model algorithms use the clustering property of graphs. We also observe that PAI works better on graphs with low GCC values. We estimate that these graphs contain many star structures and two nodes with low degrees are rarely linked by an edge. The large number of claw count (28 billion) and small number of triangle count (38 k) in twitter-icwsm graph data partially confirm our estimation.

Change of graph properties

The proposed outlier edge detection algorithms are based on the clustering property of graphs. Since outlier edges are defined as edges that do not follow the clustering property, removing them should increase the coefficients that measure this property. On the other hand, some outlier edges (also called weak links in this aspect) serves an important role to connect remote nodes or nodes from different communities. Removing such edges should also extensively increase the distance of the two end nodes. Thus the coefficients that measure the distance between the nodes of a graph shall increase when outlier edges are removed. In this experiment, we verify these changes caused by the removal of the detected outlier edges.

The global clustering coefficient (GCC) and the average local clustering coefficient (ALCC) are the de facto measures of the clustering property of graphs. GCC is defined in "Comparison of outlier edge detection algorithms". Local clustering coefficient (LCC) is the ratio of the number of edges that connect neighboring nodes of a node and the number of all possible edges that connect these neighboring nodes. The LCC of node a can be expressed as

$$c_a = \frac{\left|\{\overline{ij} | i \in N_a, j \in N_a, \overline{ij} \in E\}\right|}{k_a(k_a - 1)}. \tag{29}$$

Average local clustering coefficient is the average of the local clustering coefficients of all nodes in the graph.

We use diameter, the 90-percentile effective diameter (ED) and the mean shortest path (MSP) length as distance measures between the nodes in a graph. Diameter is the

maximum shortest path length between any two nodes in a graph. 90-percentile effective diameter is the number of edges that are needed on average to reach 90% of other nodes. The mean shortest path length is the average of the shortest path length between each pair of nodes in the graph. Note, if the graph is not connected, we measure the diameter, ED and MSP of the largest component in the graph.

In this experiment, we removed 5% of the edges with the lowest authentic score. Table 4 shows the GCC, ALCC, Diameter, ED and MSP values before and after the outlier edges were removed. For comparison, we also calculated values of these coefficients after same amount of edges are randomly removed 5% from the graph.

The results show that removing the detected outlier edges clearly increases the GCC and ALCC values, while random edge removal slightly decreases the values. This confirms the enhancement of the clustering property after outlier edges are removed. The diameter, ED and MSP values all increase when the detected outlier edges were removed. This increase is much more significant than when random edges were removed. This also confirms the theoretical prediction.

Applications

In this section, we demonstrate various applications that benefit from the proposed outlier edge detection algorithms. In these applications, we use the algorithm of the PA model combined with scheme 4 and the score function defined by Eq. 23.

Impact on graph clustering algorithms

Graph clustering is an important task in graph mining [31–33]. It aims to find clusters in a graph- a group of nodes in which the number of inner links between the nodes inside the group is much higher than that between the nodes inside the group and those outside the group. Many techniques have been proposed to solve this problem [34–37].

The proposed outlier edge detection algorithms are based on the graph clustering property. They find edges that link the nodes in different clusters. These edges are also called weak links in the literature. With the proposed techniques, we can now remove detected outlier edges before applying a graph clustering algorithm. This should improve the graph clustering accuracy and reduce the computational time.

In this application, we evaluate the performance impact of the proposed outlier edge detection technique on different graph clustering algorithms. We use simulated graph data with cluster structures as used in [36, 38–40]. We generated test graphs of 512 nodes. The average degree of each node is 24. The generated cluster size varies from 16 to 256. Let d_{out} be the average number of edges that link a node from the cluster to

Table 4 Graph properties changes after noise edges removal

	Original	ER model	PA model	Random
GCC	0.111	0.121	0.120	0.105
ALCC	0.172	0.180	0.183	0.158
Diameter	18	19	20	18
ED	5.91	6.78	6.36	5.95
MSP	3.92	4.10	4.10	3.95

nodes outside the cluster. Let d be the average degree of the node. Let $\mu = \frac{d_{out}}{d}$ be the parameter that indicates the strength of the clustering structure. The smaller μ is, the stronger the clustering structure is in the graph. We varied μ from 0.2 to 0.5. Note, when $\mu = 0.5$, the graph has a very weak clustering structure, i.e. a node inside the cluster has an equal number of edges that link it to other nodes inside and outside the cluster.

We use the Normalized Mutual Information (NMI) to evaluated the accuracy of a graph clustering algorithm. The NMI value is between 0 and 1. The larger the NMI value is, the more accurate the graph clustering result is. An NMI value of 1 indicates that the clustering result matches the ground truth. More details of the NMI metric can be found in [35, 41].

We first apply graph clustering algorithms to the test graph data and record their NMI values and computational time. Then we remove 5% of the detected outlier edges from the test graph data, and apply these graph clustering algorithms again to the new graph and record their NMI values and computational time. The differences of the NMI values and the computational time show the impact of the outlier edge removal on the graph clustering algorithms.

The evaluated algorithms are LRW [42], GN [36], SLM [43], Danon [38], Louvain [34] and Infomap [44]. MCL [45] is not listed since it failed to find the cluster structure from this type of test graph data.

We repeated the experiment 10 times and calculated the average performance. Table 5 shows the NMI values before and after outlier edges were removed. The first number in each cell shows the NMI values of the clustering result on the original graph and the second number shows the NMI values of the clustering result on the graph after the outlier edges were removed.

Table 6 shows the NMI value changes in percentage. A positive value indicates that the NMI value has increased.

The results show that outlier edge removal improves the accuracy of most graph clustering algorithms. The clustering accuracy of the SLM algorithm and the Louvain algorithm decrease slightly in some cases.

Table 7 shows the computational time changes in percentage before and after outlier edges are removed. Negative values indicate that the computational time is decreased.

These results show that outlier edge removal decreases the computational time of most algorithms used in the experiment. In some cases, SLM and the Louvain algorithms show significant gains in computation time. Note further that the increase of the

Table 5 The NMI values before and after outlier edges were removed

μ	LRW	GN	SLM	Danon	Louvain	Infomap
0.2	1.0/1.0	0.99/1.0	1.0/1.0	0.99/1.0	1.0/1.0	1.0/1.0
0.25	0.97/1.0	0.98/0.99	1.0/1.0	0.99/0.98	1.0/1.0	1.0/1.0
0.3	0.89/0.95	0.93/0.97	1.0/1.0	0.95/0.98	1.0/1.0	0.92/1.0
0.35	0.78/0.82	0.74/0.72	0.96/0.94	0.66/0.84	0.90/0.86	0.36/0.91
0.4	0.80/0.86	0.66/0.70	0.83/0.81	0.67/0.70	0.84/0.81	0.78/0.83
0.45	0.25/0.73	0.53/0.52	0.71/0.67	0.51/0.55	0.68/0.60	0.22/0.43
0.5	0.03/0.61	0.39/0.47	0.58/0.56	0.39/0.49	0.51/0.53	0/0.47

Table 6 Changes of normalized mutual information on graph clustering algorithms in percentage

μ	LRW	GN (%)	SLM	Danon (%)	Louvain	Infomap
0.2	0	0.8	0	1.0	0	0
0.25	3.3%	1.5	0	−1.0	0	0
0.3	7.3%	5.0	0	3.5	0	9.1%
0.35	5.7%	−2.2	−2.1	26	−4.9%	155%
0.4	8.5%	6.7	−2.2	4.8	−3.0%	5.8%
0.45	190%	−1.1	−6.2	8.4	−12%	95%
0.5	1730%	19	−4.4	26	2.4%	∞

Table 7 Changes of computational time on graph clustering algorithms in percentage

μ	LRW (%)	GN (%)	SLM (%)	Danon (%)	Louvain (%)	Infomap (%)
0.2	−52	−11	−36	−3.1	−33	−47
0.25	−23	−18	1.0	−1.0	−41	−16
0.3	−8.9	−9.3	7.7	−1.4	−31	−13
0.35	−11	−0.3	−21	−3.5	−35	31
0.4	−11	−5.7	−5.3	−3.0	−20	17
0.45	−16	2.8	−14.4	2.1	−41	33
0.5	−21	−6.7	−1.9	−3.4	−39	55

computational time in the Infomap algorithm leads to a crucial improvement of the clustering accuracy.

Outlier node detection in social network graphs

As mentioned in "Previous work", many algorithms have been proposed to detect outlier nodes in a graph. In this section we present a technique to detect outlier nodes using the proposed outlier edge detection algorithm.

In a social network service, if a user generates many links that do not follow the clustering property, we have good reasons to suspect that the user is a scammer. To detect this type of outlier nodes, we can first detect outlier edges. Then we find nodes that are the end points of these outlier edges. Nodes that are linked to many outlier edges are likely to be outlier nodes.

In this application, we use Brightkite data for outlier node detection. In the experiment, we rank the edges according to their authentic scores. We take the first 1000 edges as outlier edges and rank each node according to the number of outlier edges that it is connected to.

Table 8 shows the top 8 detected outlier nodes: the node ID, the number of outlier edges that the node links, the degree of the node, the rank of the degree among all nodes and LCC values of the node.

The results show that the detected outlier nodes tend to have large degree values. In particular, the LCC values of the detected outlier nodes are extremely low comparing to the ALCC value (0.172) of the graph. This shows that the neighboring nodes of the detected outlier nodes have very weak clustering property.

Table 8 Outlier node detection results on Brightkite graph

Node id	Outlier edges	Degree	Degree rank	LCC
41	21	1134	1	0.005
458	16	1055	2	0.001
115	9	838	4	0.004
175	7	270	39	0.001
989	7	270	40	0.015
2443	7	379	16	0.010
36	5	467	11	0.005
158	5	833	5	0.004

Clustering of noisy data

Clustering is one of the most important tasks in machine learning [46]. During the last decades, many algorithms have been proposed, i.e. [47–49]. The task becomes more challenging when noise is present in the data. Many algorithms, especially connectivity-based clustering algorithms, fail over such data. In this section we present a robust clustering algorithm that uses the proposed outlier edge detection techniques to find correct clusters in noisy data.

Graph algorithms have been successfully used in clustering problems [50, 51]. To cluster the data, we first build a mutual k-nearest neighbor (MKNN) graph [52, 53]. Let $x_1, x_2, \ldots, x_n \in R^d$ be the data points, where n is the number of data points and d is the dimension of the data. Let $d(x_i, x_j)$ be the distance between two data points x_i and x_j. Let $N_k(x_i)$ be the set of data points that are the k-nearest neighbors of the data point x_i with respect to the predefined distance measure $d(x_i, x_j)$. Therefore, the cardinality of the set $N_k(x_i)$ is k. A MKNN graph is built in the following way. The nodes in the MKNN graph are the data points. Two nodes x_i and x_j are connected if $x_i \in N_k(x_j)$ and $x_j \in N_k(x_i)$. The constructed MKNN graph is unweighted and undirected.

With a proper distance function, data points in a cluster are close to each other whereas data points in different clusters are far away from each other. Thus, in the constructed MKNN graph, a node is likely to be linked to other nodes in the same cluster while the links between the nodes in different clusters are relatively less. This indicates that the MKNN graph has the clustering property similar to social network graphs.

Outlier data points are normally far away from the normal data points. Some outlier nodes form isolated small components in the MKNN graph. However, the outlier nodes that fall between the clusters form bridges that connect different clusters. These bridges greatly degrade the performance of connectivity-based clustering algorithms, such as single-linkage clustering algorithm and complete-linkage clustering algorithm [46].

Based on these observations, we propose a hierarchical clustering algorithm by iteratively removing edges (weak links) according to their authentic scores. When a certain amount of outlier edges is removed, different clusters form separate large connected components—a connected component in a graph that contains a large proportion of the nodes, and it is straightforward to find them in the graph. A breadth-first search or a depth-first search algorithm can find all connected components in a graph with the complexity of $O(n)$, where n is the number of nodes. At each iteration step, we find large

connected components in the MKNN graph and the data points that do not belong to any large connected components are classified as outliers.

Using the proposed algorithm, we cluster a dataset taken from [54]. Figure 4 shows some results of different number of detected clusters. Outliers are shown in light gray color and data points in different clusters are shown in different colors.

As the Fig. 4 shows, the proposed algorithm cannot only classify outliers and normal data points but also find clusters in the data points. As more and more edges are removed from the MKNN graph, the number of clusters increases.

Next we show how to determine the true number of clusters. Table 9 shows the number of removed edges and the number of detected clusters of this dataset.

As the result shows, removing a small amount of edges is enough to find correct clusters in the data. One has to remove a large amount of edges to break a genuine cluster into smaller components. We can simply define a threshold and stop the iteration if the number of clusters does not increase any more.

To illustrate the performance of the proposed clustering algorithm, we use synthetic data that are both noisy and challenging. Figure 5 shows the test datasets. We used tools from [55] to generate the normal data points and added random data points as noise.

In our experiments, we use the Euclidean distance function. The number of nearest neighbors is 30. At each iteration step, we remove 0.1% of total number of edges according to their authentic scores. A large connected component is a component whose size is

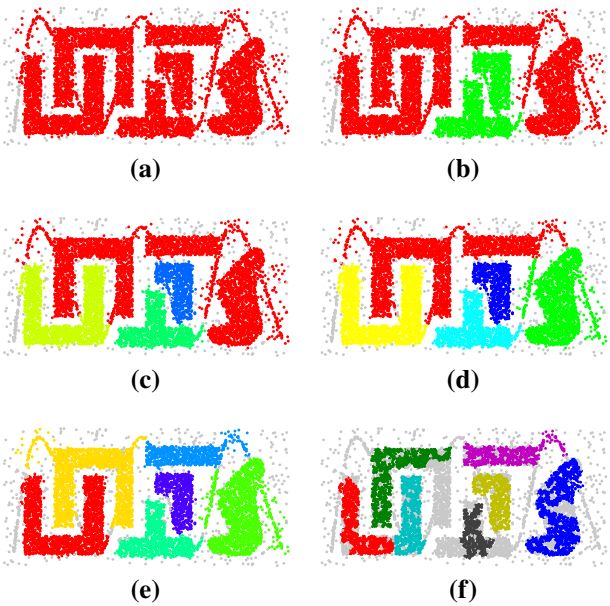

Fig. 4 Given the number of clusters, the clustering results of a dataset taken from [54]. **a** 1 cluster; **b** 2 clusters; **c** 4 clusters; **d** 5 clusters; **e** 6 clusters; **f** 7 clusters

Table 9 Percentage of the removed edges and the number of detected clusters

Removed edges	2.6%	2.7%	2.8%	3.5%	6%	33.3%
Number of clusters	2	3	4	5	6	7

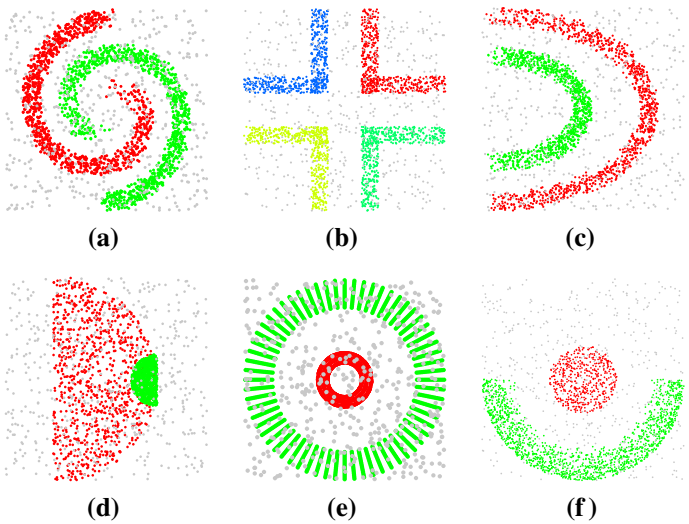

Fig. 5 Synthetic datasets for clustering. Clusters are separated by different *colors* and noise data is shown by *grey dots*. **a** two spirals; **b** corners; **c** half kernels; **d** unbalanced densities; **e** cluster in cluster; **f** crescent and full moon

larger than 5% of the total number of nodes. The clustering termination threshold is set as 10% of the total number of edges.

We compare the proposed clustering algorithm with the k-means[46], the average-linkage (a-link) [46], the normalized cuts (N-Cuts) [56] and the graph degree linkage (GDL) [49] clustering algorithms. Since the competing algorithms cannot detect the number of clusters, we use the value from the ground truth. Table 10 shows the NMI scores of the proposed algorithm and the competing algorithms.

The results show that the k-means and the average linkage clustering algorithms fail on complex-shaped clusters. GDL and the proposed algorithms are all graph-based clustering algorithms. They are able to find clusters with arbitrary shapes. From the NMI scores, the proposed algorithm is clearly superior to the competing clustering algorithms.

Conclusions

In real-world graphs, in particular social network graphs, there are edges.

generated by scammers, malicious programs or mistakenly by normal users and the system. Detecting these outlier edges and removing them will not only improve the efficiency of graph mining and analytics, but also help identify harmful entities. In this article, we introduce outlier edge detection algorithms based on two random graph

Table 10 Clustering of noisy data results

Dataset	k-Means	a-Link	N-Cuts	GDL	Proposed
(a)	0.031	0.099	0.053	0.650	*0.672*
(b)	0.743	0.743	0.743	0.743	*0.848*
(c)	0	0.004	0.559	0.654	*0.755*
(d)	0.208	0.161	0.367	0.553	*0.619*
(e)	0.001	0.133	0.680	0.701	*0.744*
(f)	0.001	0.162	0.627	0.612	*0.714*

Italics indicates the best score of each experiment

generation models. We define four schemes that represent relationships of two nodes and the groups of their neighboring nodes. We combine the schemes with the two random graph generation models and investigate the proposed algorithms theoretically. We tested the proposed outlier edge detection algorithms by experiments on real-world graphs. The experimental results show that our proposed algorithms can effectively identify the injected edges in real-world graphs. We compared the performance of our proposed algorithms with other outlier edge detection algorithms. The proposed algorithms, especially the algorithm based on the PA model, give consistently good results regardless of the test graph data. We also evaluated the changes of graph properties caused by the removal of the detected outlier edges. The experimental results show an increase in both the clustering coefficients and the increase of the distance between the nodes in the graph. This is coherent with the theoretical predictions.

Further more, we demonstrate the potential of the outlier edge detection using three different applications. When used with the graph clustering algorithms, removing outlier edges from the graph not only improves the clustering accuracy but also reduces the computational time. This indicates that the proposed algorithms are powerful preprocessing tools for graph mining. When used for detecting outlier nodes in social network graphs, we can successfully find outlier nodes whose behavior deviates dramatically from that of normal nodes. We also present a clustering algorithm that is based on the edge authentic scores. The clustering algorithm can efficiently find true data clusters by excluding noises from the data.

Outlier edge detection has great potentials in numerous Big Data applications. In the future, we will apply the proposed outlier edge detection algorithms in applications in other fields, for example computer vision and content-based multimedia retrieval in the Big Visual Data. We observed that nodes and edges outside edge-ego-network also contain valuable information in outlier detection. However, using this information dramatically increases the computational cost. We will work on fast algorithms that can efficiently use the structural information of the whole graph.

Abbreviations

ALCC: average local clustering coefficient; AUC: area under the curve; CN: common neighbors; ED: effective diameter; ER: Erdős- Rényi; GCC: global clustering coefficient; GDL: graph degree linkage; GN: Girvan–Newman; HDI: hub depressed index; HPI: hub promoted index; LCC: local clustering coefficient; LRW: limited random walk; MC: mean conductance; MCL: Markov Clustering Algorithm; MKNN: mutual k-nearest neighbor; MSP: mean shortest path; NMI: Normalized Mutual Information; N-Cuts: normalized cuts; PA: preferential attachment; PAI: preferential attachment index; ROC: receiver operating characteristic; SLM: smart local moving.

Authors' contributions

HZ carried out the conception and design of the study, participated in the analysis and interpretation of data, and was involved in drafting and revising the manuscript. SK and MG made substantial contributions to the design of the study, the analysis and interpretation of the data, and were involved in critically reviewing the manuscript. All authors read and approved the final manuscript.

Author details

[1] Department of Signal Processing, Tampere University of Technology, Finland, Korkeakoulunkatu 1, FI-33101 Tampere, Finland. [2] Electrical Engineering Department, College of Engineering, Qatar University, Qatar, 2713, Al Hala St, Doha, Qatar.

Competing interests

The authors declare that they have no competing interests.

Funding

Achademy of Finland supported this research.

Appendix: Proof of Theorem 2

Proposition 6.1 $\alpha(S \cup T, R) = \alpha(S, R) + \alpha(T, R)$ *if* $S \cap T = \emptyset$.

Proof Let A be the adjacency matrix of an unweighted and undirected graph G. We have $\alpha(S, T) = \sum_{i \in S} \sum_{j \in T} A_{ij}$. Given $S \cap T = \emptyset$,

$$\begin{aligned}\alpha(S \cup T, R) &= \sum_{i \in S \cup T} \sum_{j \in R} A_{ij} \\ &= \sum_{i \in S} \sum_{j \in R} A_{ij} + \sum_{i \in T} \sum_{j \in R} A_{ij} \\ &= \alpha(S, R) + \alpha(T, R)\end{aligned}$$

Next we prove Theorem 2.

Proof For scheme 4, $P^{(4)}_{a,b} = S_{a \setminus b}$, $R^{(4)}_{a,b} = S_{b \setminus a}$, $P^{(4)}_{b,a} = S_{b \setminus a}$ and $R^{(4)}_{b,a} = S_{a \setminus b}$. Using Theorem 1, we can easily get $\alpha\left(P^{(4)}_{a,b}, R^{(4)}_{a,b}\right) = \alpha\left(P^{(4)}_{b,a}, R^{(4)}_{b,a}\right)$.

To prove Theorem 2 for scheme 2, we divide the nodes in edge-ego-network $G_{\overline{ab}}$ into five mutually exclusive sets:

- $V_1 = \{x | x \in N_a \text{ and } x \notin S_b\}$;
- $V_2 = \{x | x \in N_b \text{ and } x \notin S_a\}$;
- $V_3 = \{x | x \in N_a \text{ and } x \in N_b\}$;
- $V_4 = \{a\}$;
- $V_5 = \{b\}$.

From the definition, we have

$$\begin{aligned}P^{(2)}_{a,b} &= N_{a \setminus b} = V_1 \cup V_3, \\ R^{(2)}_{a,b} &= S_{b \setminus a} = V_2 \cup V_3 \cup V_5, \\ P^{(2)}_{b,a} &= N_{b \setminus a} = V_2 \cup V_3, \\ R^{(2)}_{b,a} &= S_{a \setminus b} = V_1 \cup V_3 \cup V_4.\end{aligned}$$

Using the definition of $\alpha(S, T)$ and Proposition 6.1, we get

$$\begin{aligned}\alpha\left(P^{(2)}_{a,b}, R^{(2)}_{a,b}\right) &= \alpha(V_1 \cup V_3, V_2 \cup V_3 \cup V_5) \\ &= \alpha(V_1, V_2) + \alpha(V_1, V_3) + \alpha(V_1, V_5) \\ &\quad + \alpha(V_3, V_2) + \alpha(V_3, V_3) + \alpha(V_3, V_5)\end{aligned} \tag{30}$$

and

$$\begin{aligned}\alpha\left(P^{(2)}_{b,a}, R^{(2)}_{b,a}\right) &= \alpha(V_2 \cup V_3, V_1 \cup V_3 \cup V_4) \\ &= \alpha(V_2, V_1) + \alpha(V_2, V_3) + \alpha(V_2, V_4) \\ &\quad + \alpha(V_3, V_1) + \alpha(V_3, V_3) + \alpha(V_3, V_4)\end{aligned} \tag{31}$$

Taking the fact that $\alpha(V_1 \cap V_5) = 0$, $\alpha(V_2 \cap V_4) = 0$, and $\alpha(V_3, V_4) = \alpha(V_3, V_5)$, the right hand side of Eqs. 30 and 31 are equal. Thus $\alpha\left(P^{(2)}_{a,b}, R^{(2)}_{a,b}\right) = \alpha\left(P^{(2)}_{b,a}, R^{(2)}_{b,a}\right)$.

Publisher's Note

Received: 8 February 2017 Accepted: 13 April 2017
Published online: 26 April 2017

References

1. Newman M. Networks: an introduction. 1st ed. New York: Oxford; 2010.
2. Jiang M, Cui P, Beutel A, Faloutsos C, Yang S. CatchSync: catching synchronized behavior in large directed graphs. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining., KDD '14. New York: ACM; 2014. p. 941–50.
3. Beutel A, Xu W, Guruswami V, Palow C, Faloutsos C. CopyCatch: stopping group attacks by spotting lockstep behavior in social networks. In: Proceedings of the 22nd international conference on World Wide Web, 2013. p. 119–130.
4. Yu R, Qiu H, Wen Z, Lin C, Liu Y. A survey on social media anomaly detection. SIGKDD Explor Newslett. 2016;18(1):1–14.
5. Akoglu L, Tong H, Koutra D. Graph based anomaly detection and description: a survey. Data Mining Knowl Discov. 2015;29(3):626–88.
6. Noble CC, Cook DJ. Graph-based anomaly detection. In: Proceedings of the Ninth ACM SIGKDD international conference on knowledge discovery and data mining. KDD '03, Washington, D.C. New York: ACM; 2003. p. 631–636. doi:10.1145/956750.956831.
7. Dai H, Zhu F, Lim EP, Pang H. Detecting anomalies in bipartite graphs with mutual dependency principles. In: 2012 IEEE 12th international conference on data mining (ICDM). IEEE; 2012. p. 171–80.
8. Henderson K, Gallagher B, Eliassi-Rad T, Tong H, Basu S, Akoglu L, Koutra D, Faloutsos C, Li L. Rolx: structural role extraction & mining in large graphs. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining. ACM; 2012. p. 1231–9.
9. Hodge VJ, Austin J. A survey of outlier detection methodologies. Artif Intell Rev. 2004;22(2):85–126.
10. Xu X, Yuruk N, Feng Z, Schweiger TAJ. SCAN: a structural clustering algorithm for networks. Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining., KDD '07. New York: ACM; 2007. p. 824–33.
11. Gao J, Liang F, Fan W, Wang C, Sun Y, Han J. On community outliers and their efficient detection in information networks. In: Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining., KDD '10. New York: ACM; 2010. p. 813–22.
12. Akoglu L, McGlohon M, Faloutsos C. oddball: spotting anomalies in weighted graphs. In: Zaki MJ, Yu JX, Ravindran B, Pudi V, eds. Advances in knowledge discovery and data mining. Lecture notes in computer science. 2010. pp. 410–421.
13. Liu L, Zuo WL, Peng T. Detecting outlier pairs in complex network based on link structure and semantic relationship. Expert Syst Appl. 2017;69:40–9.
14. Chakrabarti D. AutoPart: parameter-free graph partitioning and outlier detection. In: Boulicaut JF, Esposito F, Giannotti F, Pedreschi D, eds. Knowledge discovery in databases: PKDD 2004. Lecture notes in computer science. 2004. p. 112–24.
15. Easley D, Kleinberg J. Networks, crowds, and markets: reasoning about a highly connected world. Cambridge University Press; 2010.
16. Lu L, Zhou T. Link prediction in complex networks: a survey. Physica A Stat Mech Appl. 2011;390(6):1150–70.
17. Barbieri N, Bonchi F, Manco G. Who to follow and why: link prediction with explanations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining., KDD '14. New York: ACM; 2014. p. 1266–75.
18. Freeman LC. Centered graphs and the structure of ego networks. Math Soc Sci. 1982;3(3):291–304.
19. Watts DJ, Strogatz SH. Collective dynamics of 'small-world'networks. Nature. 1998;393(6684):440–2.
20. Coscia M, Rossetti G, Giannotti F, Pedreschi D. DEMON: a local-first discovery method for overlapping communities. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining., KDD '12. New York: ACM; 2012. p. 615–23.
21. Bollobás B. Random graphs. 2 ed. Cambridge : New York; 2001.
22. Newman MEJ. Modularity and community structure in networks. Proc Natl Acad Sci. 2006;103(23):8577–82.
23. Cho E, Myers SA, Leskovec J. Friendship and mobility: user movement in location-based social networks. In: Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining. New York: ACM; 2011. p. 1082–90.
24. Kunegis J. Konect: the koblenz network collection. In: Proceedings of the 22nd international conference on World Wide Web. New York: ACM; 2013. p. 1343–50.
25. Massa P, Salvetti M, Tomasoni D. Bowling alone and trust decline in social network sites. In: IEEE International conference on dependable, autonomic and secure computing, 2009. DASC'09 Eighth. 2009. p. 658–63.
26. De Choudhury M, Lin YR, Sundaram H, Candan KS, Xie L, Kelliher A. How does the data sampling strategy impact the discovery of information diffusion in social media? ICWSM. 2010;10:34–41.

27. Viswanath B, Mislove A, Cha M, Gummadi KP. On the evolution of user interaction in facebook. In: Proceedings of the 2nd ACM workshop on online social networks. New York: ACM; 2009. p. 37–42.
28. Leskovec J, Kleinberg J, Faloutsos C. Graph evolution: densification and shrinking diameters. ACM Transa Knowl Discov Data. 2007;1(1):2.
29. Yang J, Leskovec J. Defining and evaluating network communities based on ground-truth. Knowl Inform Syst. 2015;42(1):181–213.
30. Leskovec J, Lang KJ, Dasgupta A. Statistical properties of community structure in large social and information networks. In: Proceedings of the 17th international conference on World Wide Web. New York: ACM; 2008. pp. 695–704.
31. Fortunato S. Community detection in graphs. Phys Rep. 2010;486(3–5):75–174.
32. Coscia M, Giannotti F, Pedreschi D. A classification for community discovery methods in complex networks. Stat Anal Data Min. 2011;4(5):512–46.
33. Papadopoulos S, Kompatsiaris Y, Vakali A, Spyridonos P. Community detection in social media. Data Min Knowl Discov. 2011;24(3):515–54.
34. Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. J Stat Mech Theory Exp. 2008;2008(10):10008.
35. Danon L, Diaz-Guilera A, Duch J, Arenas A. Comparing community structure identification. J Stat Mech Theory Exp. 2005;2005(09):09008.
36. Newman ME, Girvan M. Finding and evaluating community structure in networks. Phys Rev E. 2004;69(2):026113.
37. Schaeffer SE. Graph clustering. Comput Sci Rev. 2007;1(1):27–64.
38. Danon L, Díaz-Guilera A, Arenas A. The effect of size heterogeneity on community identification in complex networks. J Stat Mech Theory Exp. 2006;2006(11):11010.
39. Lancichinetti A, Fortunato S, Radicchi F. Benchmark graphs for testing community detection algorithms. Phys Rev E. 2008;78(4):046110.
40. Newman ME. Fast algorithm for detecting community structure in networks. Phys Rev E. 2004;69(6):066133.
41. Ana LN, Jain AK. Robust data clustering. In: Proceedings 2003 IEEE computer society conference on computer vision and pattern recognition, 2003. vol. 2, p. 128–1332.
42. Zhang H, Raitoharju J, Kiranyaz S, Gabbouj M. Limited random walk algorithm for big graph data clustering. J Big Data. 2016;3(1):26.
43. Waltman L, Eck NJV. A smart local moving algorithm for large-scale modularity-based community detection. The. Eur Phys J B. 2013;86(11):1–14.
44. Rosvall M, Bergstrom CT. Maps of random walks on complex networks reveal community structure. Proc Natl Acad Sci. 2008;105(4):1118–23.
45. Dongen S. Graph clustering by flow simulation. PhD thesis, Utrecht: Universiteit Utrecht; 2000.
46. Theodoridis S, Koutroumbas K. Pattern recognition. 4 ed. Amsterdam; 2008.
47. Jain AK, Murty MN, Flynn PJ. Data clustering: a review. ACM Comput Surv. 1999;31(3):264–323.
48. Lloyd S. Least squares quantization in PCM. IEEE Trans Inform Theory. 1982;28(2):129–37.
49. Zhang W, Wang X, Zhao D, Tang X. Graph degree linkage: agglomerative clustering on a directed graph. In: Fitzgibbon A, Lazebnik S, Perona P, Sato Y, Schmid C, editors. Computer Vision - ECCV 2012. Lecture Notes in Computer Science. 2012. p. 428–41.
50. Harel D, Koren Y. On clustering using random walks. In: Hariharan R, Vinay V, Mukund M, editors. FST TCS 2001: Foundations of software technology and theoretical computer science. Lecture notes in computer science. 2001. pp. 18–41.
51. Dong X, Frossard P, Vandergheynst P, Nefedov N. Clustering with multi-layer graphs: a spectral perspective. IEEE Trans Sign Process. 2012;60(11):5820–31.
52. Brito MR, Chávez EL, Quiroz AJ, Yukich JE. Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection. Stat Probab Lett. 1997;35(1):33–42.
53. Ozaki K, Shimbo M, Komachi M, Matsumoto Y. Using the mutual k-nearest neighbor graphs for semi-supervised classification of natural language data. In: Proceedings of the fifteenth conference on computational natural language learning. 2011. p. 154–62.
54. Karypis G, Han E-H, Kumar V. Chameleon: hierarchical clustering using dynamic modeling. Computer. 1999;32(8):68–75.
55. 6 functions for generating artificial datasets - File Exchange - MATLAB Central. http://se.mathworks.com/matlabcentral/fileexchange/41459. Accessed 23 Feb 2017.
56. Shi J, Malik J. Normalized cuts and image segmentation. IEEE Trans Pattern Anal Mach Intell. 2000;22(8):888–905.