Incremental Local Evolutionary Outlier Detection for Dynamic Social Networks

Tengfei Ji, Dongqing Yang, and Jun Gao

School of Electronics Engineering and Computer Science,
Peking University, Beijing, 100871 China
tfji@pku.edu.cn

Abstract. Numerous applications in dynamic social networks, ranging from telecommunications to financial transactions, create evolving datasets. Detecting outliers in such dynamic networks is inherently challenging, because the arbitrary linkage structure with massive information is changing over time. Little research has been done on detecting outliers for dynamic social networks, even then, they represent networks as un-weighted graphs and identify outliers from a relatively global perspective. Thus, existing approaches fail to identify the objects with abnormal *evolutionary behavior only* with respect to their *local neighborhood*. We define such objects as local evolutionary outliers, LEOutliers. This paper proposes a novel *incremental* algorithm IcLEOD to detect LEOutliers in *weighted* graphs. By focusing only on the time-varying components (*e.g.*, node, edge and edge weight), IcLEOD algorithm is highly efficient in large and gradually evolving networks. Experimental results on both real and synthetic datasets illustrate that our approach of finding local evolutionary outliers can be practical.

Keywords: Outlier detection, Dynamic Social Networks, Weighted evolving graphs, Local information.

1 Introduction

Outlier detection is a task to uncover and report observations which appear to be inconsistent with the remainder of that set of data [1]. Since outliers are usually represented truly unexpected knowledge with underlying value, research has been widely studied in this area, often applicable to static traditional strings or attribute-value datasets [2].

Little work, however, has focused on outlier detection in dynamic graph-based data. With the unprecedented development of social networks, various kinds of records like credit, personnel, financial, medical, etc. all exist in a graph form, where vertices represent objects, edges represent relationships among objects and edge weights represent link strength [3]. Graph-based outlier detection problem is specially challenging for three major reasons as follows:

Dynamic Changes: Vertices, the relationships among them as well as the weight of the relationships are all continuously evolving. For example, users join friendship networks (*e.g.* Facebook), friendships are established, and communication

H. Blockeel et al. (Eds.): ECML PKDD 2013, Part II, LNAI 8189, pp. 1–15, 2013.

becomes increasingly frequent. To capture outliers in evolving networks, detecting approaches should obtain temporal information from a collection of snapshots instead of a particular instant. For example, snapshots of the Facebook graph should be taken periodically, forming a sequence of snapshot graphs [4].

Massive Information: Compared with average data sets, social networks are significantly larger in size. The volume is even larger when the network is dynamic, massive information involved in a series of snapshots with millions of nodes and billions of edges[5]. In this case, it is difficult for algorithms to obtain full knowledge of the entire networks.

Deeply Hidden Outliers: Recent studies suggest that social networks usually exhibit hierarchical organization, in which vertices are divided into groups that can be further subdivided into groups of groups, and so forth over multiple scales [21]. Therefore, outliers are more difficult to distinguish from normal ones if they are hidden deeply among their neighbored but not globally.

However, outlier detection in social networks has not yet received as much attention as some other topics, e.g. community discovery [9,10]. Only a few studies have been conducted on graph-based outlier detection (e.g. [3], [6], [7], [8]). While a more detailed discussion on these approaches will be provided in section 2, it suffices to point out here that most of these approaches identify outliers in *unweighted* graphs from a more *global* perspective. For example, community-based algorithms [3,6] identify objects whose evolving trends are different with that of entire community. All such global outlier detection algorithms require the entire structure of the graph be fully known, which is impractical when dealing with large evolving networks. Furthermore, the local abnormality may be highly covered by global evolution trend. Thus, existing global methods fail to identify the objects with abnormal evolutionary behavior *only* relative to their *local neighborhood*. We define such objects as local evolutionary outliers, LEOutliers. The following example is adopted to illustrate directly the feature of LEOutliers.

Example: Who Should be Liable for Examination Leakage

Figure 1 shows a communication network with two communities, teacher community C_1 and student community C_2. Different colors are used to distinguish between members of two communities. Because of space constraints, links between nodes have been omitted. It is worthwhile to note that we use the **overlapping area** of two communities to denote the interactions between teachers and students. The more they are connected, the larger the overlapping area becomes.

Figure 1(a) contains two snapshots at time T-1 and T and we suppose that the Entrance Examination time is near at T. It is obvious that, from T-1 to T, the evolution trend of entire teacher community is communicating more frequently with student community, which is reasonable since more guidance is needed before examination. According to the global-view algorithms, objects that follow the entire community evolution trend are regarded as normal ones. Interestingly, once local neighborhood is taken into account, as illustrated in Figure 1(b), the black node v is an example of local evolutionary outlier. We suppose v

and its neighbors at time T-1 (blue triangles) are a special kind of teachers, paper setters. The blue triangles avoid communicating with students as the examination approaches for confidential reasons. On the contrary, node v is behaving abnormally as he frequently interacts with students at T, which is a violation of principle. Therefore, although node v evolving consistently with entire community, he is the most likely suspect in examination leakage.

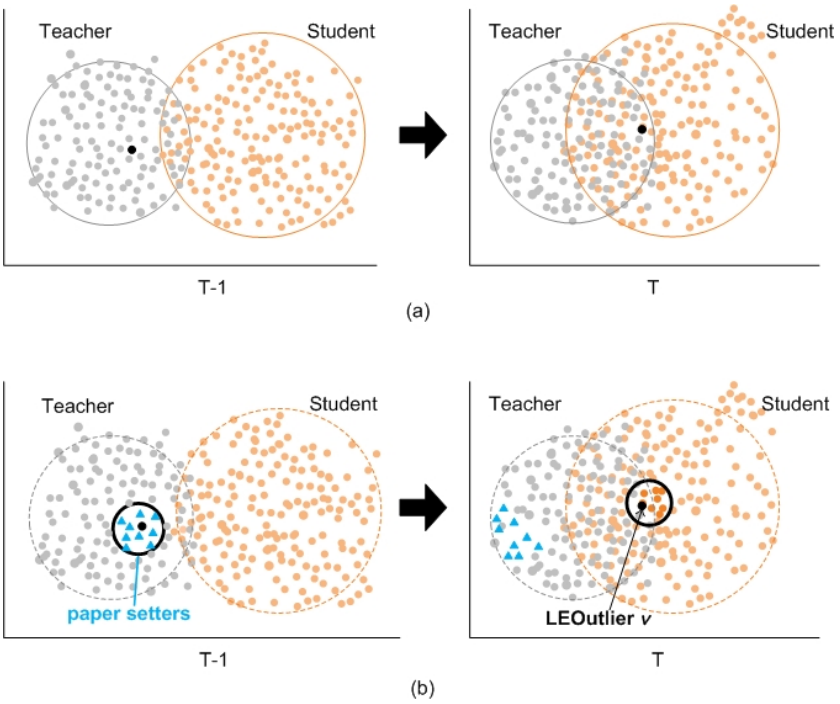

Fig. 1. Example of LEOutlier

The above example shows that the global-view algorithm is adequate under certain conditions, but not satisfactory for the case when evolutionary outliers are hidden deeply among their neighborhood. In this paper, we propose a novel method named IcLEOD to effectively detect LEOutlier in weighted graphs from a local perspective. The technical contributions of our work can be summarized as follows:

- Besides descriptive concept, we put forward a novel measurable definition of *local evolutionary outlier*. To the best of our knowledge, this is the first straightforward concept of a local evolutionary outlier which quantifies how outlying an object's evolving behavior is from a local perspective.
- We propose an incremental local evolutionary outlier detection algorithm (IcLEOD), which fully considers the varying temporal information and the

complex topology structure of social networks. Our algorithm consists of two stages: In stage I, a local substructure named $Corenet(v)$ is constructed for every object v according to structure information and edge weights; In stage II, we detect local evolutionary outliers by carefully analyzing and comparing the $Corenet(v)$ at different snapshots.
- Our algorithm greatly increases the efficiency by incrementally analyzing the dynamic components (e.g., node, edge and edge weight) and the limited number of nodes affected by them. This technique is more favorable than algorithms that require global knowledge of the entire network, especially in the case that the snapshot graphs are gradually evolving.
- Finally, the extensive experiments on both real and synthetic datasets confirm the capability and the performance of our algorithm. We conclude that finding local evolutionary outliers using IcLEOD is meaningful and practical.

The rest part of this work is organized as follows: Section 2 discusses the recent related work; Section 3 proposes our incremental local evolutionary outlier detection algorithm, IcLEOD; Section 4 gives experiments for our approach on both real and synthetic data sets, and shows the achieved results. Section 5 makes a conclusion about the whole work.

2 Related Work

To focus on the theme, the traditional non-graph based outlier detection algorithms will no more be introduced in this paper (e.g., distance-based [17], distribution-based [1] and density-based methods [15,16]). We are eager to discuss some state-of-the-art algorithms that conduct on graphs. Graph-based outlier detection has been studied from two major perspectives: global versus local. We will introduce some typical methods in both categories respectively.

Graph-Based *Global* Outlier Detection Methods: Most recent work on graph-based outlier detection has focused on unweigted graphs from a more global perspective (i.e. entire graph, community). For example, a stream-based outlier detection algorithm [14] takes a global view of entire graph to identify graph objects which contain unusual bridging edges. Community-based outlier detection methods [7,13] detect outliers within the context of communities such that the identified outliers deviate significantly from the rest of the community members. Some methods [3,6] capture the dynamic anomalous objects whose evolution behaviors are quite different from that of their respective communities. All global outlier detection algorithms require that the entire graph should be obtained, which may be impractical if networks are too large or too dynamic.

Graph-Based *Local* Outlier Detection Methods: Saligrama [11] proposes a statistical method based on local K-nearest neighbor distances to identify anomalies localized to a small spatial region, which is used mainly to deal with spatial data and cannot be easily generalized to non-spatial networks. OddBall algorithm [12] takes a egocentric view to search weighted graphs based upon a

set of power laws, and determines four types of anomalous subgraphs centered on individual nodes: near-cliques, near-stars, heavy vicinities and dominant heavy links. Los Alamos National Laboratory [20] explores local areas and paths in the network which are least likely to occur under normal conditions by combining anomaly scores from edges in a neighborhood. Most methods in this category utilize only single snapshot data to find unexpected nodes/edges/sub-structures and hence they cannot detect temporal changes.

In summary, most of existing methods represent social networks (static and dynamic) as unweighted graphs, and find outliers from a global point of view. Thus the outliers detected by previous algorithms are not local evolutionary outliers as proposed in this paper.

3 IcLEOD Algorithm

Consider a dynamic social network as a sequence of snapshots G_1, G_2, ...,G_T, each snapshot is represented by weighted graphs G = (V, E), where V is the set of objects (nodes) and E is the set of weighted edges. The weight of an edge denotes the link strength (connecting times). In this paper, we focus on the problem of detecting local evolutionary outliers from any of the two snapshots G_i and G_j. Local evolutionary outliers across multiple snapshots can be obtained by simple post-processing. More specifically, input for our problem thus consists two snapshots of a weighted evolving network, and meaningful LEOutliers are output.

Our LEOD algorithm involves two major phases. In the first phase, *Corenet* for individual object is formed according to local topology structure and edge weights information. In the second phase, local evolutionary outliers are identified by comparing individual's *Corenet*s of different snapshots. We will present two phases in Subsection 3.1 and 3.2 respectively.

3.1 Phase I: Discovering Corenet for Individual Object

As noted above, the evolutionary behavior of a LEOutlier is extremely different from that of its "closest" neighbors. Thus, the primary goal in phase I is to reasonably measure the closeness between objects, so as to determine which nodes could be regarded as the closest ones. There are two basic concepts usually used to group local nodes in un-weighted graph [19]. We will briefly introduce them before providing the notion of *Corenet*.

Definition 1 (Egonet). Given a node(ego) $v_i \in$ V, the egonet of v_i is defined as egonet(v_i)=$\{v_i\} \cup \{v_j \mid v_j \in V, e_{ij} \in E\}$

Where e_{ij} is the edge between v_i and v_j.

Definition 2 (Super-egonet). Given a node(ego) $v_i \in$ V, the super-egonet of v_i is defined as super-egonet(v_i)=$\{ego(v_i)\} \cup \{ego(v_j) \mid v_j \in V, e_{ij} \in E\}$

Obviously, these two concepts are very simple in obtaining the local substructure: they just regard 1-hop neighbors(egonet) or neighbors within 2-hop(super-egonet) as the ego's closest neighbors. However, they will encounter problems

when dealing with weighted graphs. As in the case of a friendship network with edge-weights representing interactions between friends, one is likely to be closer to his intimate friend's intimate friend instead of his nodding acquaintances. Consider the situation in Figure 2, where node X is the ego, Y_1, Y_2,Y_3 are 1-hop neighbors of X, Z_1 is its 2-hop neighbor. By following the definition of egonet, as Figure 2(b) shows, Y_1, Y_2 and Y_3 are the 3-closest neighbors of X. The concept of egonet focuses only on structural connection but ignores the power of closeness transmission. Therefore, it requires a forceful measurement considering both *connectivity* and *closeness*.

First, we propose the following two notions to assess the *closeness* between ego and its neighbors. We call the node of interest *core* to differentiate it from egonet.

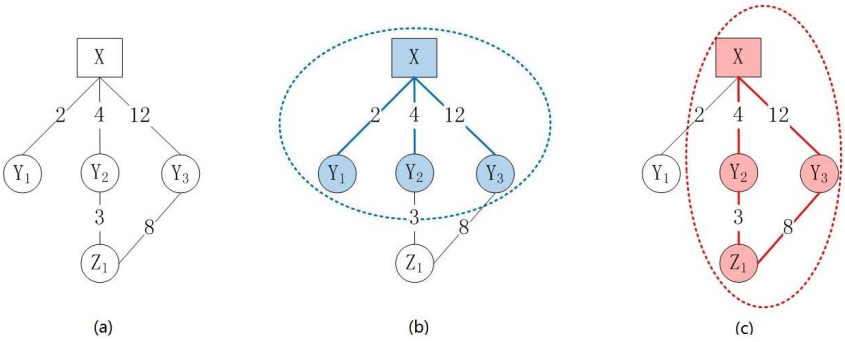

Fig. 2. Comparison of Egonet and Corenet

Definition 3 (Closeness related to the core). Let node v_0 be core, $v_0 \in$V. For $\forall v_l \in V$, we assume that there are d paths connecting v_0 and v_l. The jth path (l in length) passes through nodes $\{v_0, v_1, v_2, ..., v_l\}$ in sequence, where $1 \leqslant j \leqslant d$. Then the closeness between v_0 and v_l is defined as:

$$Closeness(v_0, v_l) = \max_{1 \leq j \leq d} \prod_{i=0}^{l-1} \frac{w_{v_i v_{i+1}}}{w_{v_i}} \tag{1}$$

Where $w_{v_i v_{i+1}}$ is the weight of the edge between v_i and v_{i+1}, and w_{v_i} is the sum of the weights of the edges connected to node v_i. Obviously, $\forall v_l \in V$, $Closeness(v_0, v_l) \in [0, 1]$. The higher the value, the more intimate the relation is. It is possible that a node directly connected with the core owns a smaller closeness. For example, in Figure 2, $Closeness(X, Y_1) = \frac{2}{2+4+12} = \frac{1}{9}$ and $Closeness(X, Z_1) = \frac{12}{2+4+12} \times \frac{8}{12+8} = \frac{4}{15}$.

In the case that two (or more) identical values of closeness are obtained from two (or more) different paths, to avoid closeness drift, we prefer the path that includes the edge directly connecting the core with maximum weight.

Definition 4 (k-closeness of the core). Let node v_0 be core, $v_0 \in$V. For $\forall$ k>0, the k-closeness of the core, denoted as k-closeness(v_0), is defined as :

(i) For at least k nodes $v_p \in$ V\{v_0}, it holds that Closeness(v_0,v_p)$\geq$ k-closeness(v_0), and

(ii) For at most k-1 nodes $v_p \in$ V\{v_0}, it holds that Closeness(v_0,v_p)$>$ k-closeness(v_0).

Different with the concepts of Egonet and Super-egonet, the definition 4 considers the top-k "closest" neighbors of the core only based on closeness transmission, instead of linking relationships. In this definition, the "closest" neighbors are those nodes with larger value of closeness, rather than directly connecting with the core.

Definition 5 (k-closeness neighborhood of the core). Given the k-closeness of core v_0, the k-closeness neighborhood of v_0 contains every node whose closeness related to v_0 is not smaller than the k-closeness(v_0). Formally, $N_k(v_0) = \{v_p \in$ V\{v_0} | Closeness(v_0,v_p) $\geq$ k-closeness(v_0)$\}$.

As mentioned above, egonet concerns only the nodes directly connected with the node of interest, while the closeness measurement (Def. 3-5) mainly consider closeness transmission. The former completely ignores the edge-weight information, similarly, the latter ignores the risk that the reliability may reduce after successive transmissions. Thus, for the purpose of discovering the local context for the core, we propose a notion named *Corenet* that balances the topology structure and the closeness transmission.

Definition 6 (Corenet). Given the k-closeness of core, k-closeness(v_0), the Corenet of v_0 contains nodes that satisfy the conditions: (i) the closeness related to v_0 is not smaller than the k-closeness(v_0), and (ii) they are in the super-egonet of v_0. Formally, $v_p \in$ super-egonet(v_0)\{v_0}, Corenet(v_0) is defined as:

$$\mathrm{Corenet}(\mathrm{v_0}) = \begin{cases} \mathrm{super-egonet(v_0)}, & \textit{if } \min\limits_{\mathrm{v_p}} \mathrm{Closeness(v_0, v_p)} \geq \mathrm{k-closeness(v_0)} \\ \mathrm{N_k(v_0)}, & \text{others} \end{cases}$$

So far, we have defined *corenet* as the local context of the core, which fully takes closeness transmission into account and avoids meaningless excessive transmissions by imposing a structural restriction. It is obvious that only the nodes in super-egonet(v_0) need to be calculated closeness related to the core and the maximum size of corenet is the number of the core's neighbors within 2-hop.

3.2 Phase II: Measuring Outlying Score

In this subsection, we will discuss how to detect LEOutliers by comparing *Corenets* at different snapshots. Since most real social networks are gradually evolving, which means successive snapshots are likely similar to each other (sharing more than 99% of their edges [4]). We utilize this property to exploit redundancies among similar snapshots and focus only on the components changing

over time. The time-varying components and their notations are listed in Table 1. The changes of these components will affect their neighbors in a certain range. For example, if Z_1 is deleted from Figure 2(c), it will affect the *Corenet* of X. Thus, the *Corenet* of X need to be redetermined and X has to be examined for any anomalous evolving behavior. The following definition describes the influence of the time-vary components.

Table 1. Time-varying Components and their Notations

Time-varying Component	Event at time t+1	Symbol
Node	insertion of a new object	v^+
	deletion of an old object	v^-
Edge	generation of a new edge	e^+ with endpoints v_{e^+}
	deletion of an old edge	e^- with endpoints v_{e^-}
Edge-weight	increase weight of an edge	w^+ with endpoints v_{w^+}
	decrease weight of an edge	w^- with endpoints v_{w^-}

Definition 7 (Incremental nodes collection: *IC*). Given two snapshots G_{T-1} and G_T, the differences between them are time-vary components, as illustrated in Table 1. The range of nodes that could be affected by time-varying components is defined as:

$$\begin{aligned} IC = &\ superegonet^T(v^+) \cup superegonet^{T-1}(v^-) \\ &\cup egonet^T(v_{e^+}) \cup egonet^{T-1}(v_{e^-}) \\ &\cup egonet^T(v_{w^+}) \cup egonet^{T-1}(v_{w^-}) \end{aligned}$$

Where $superegonet^T(v^+)$ is the super-egonet of time-varying node v^+ in graph G_T, and other five are similar.

From definition 7, the time-vary components influence only limited number of their neighbors, namely nodes in *IC*. Thus our algorithm only need to examine the nodes in *IC* instead of the total number of nodes in the social network.

Before we present the particular measuring function, we first analyze the signs that a node is evolving abnormally. Consider we have two snapshots G_{T-1} and G_T, and the node of interest is v, there are two major signs to show that v is likely to be a LEOutlier:

(1) The members of Corenet(v) in G_{T-1} no longer belong to Corenet(v) or their closeness related to v is getting weaker from G_{T-1} to G_T;

(2) The new members added to Corenet(v) at time T have clear distinction with the former members, moreover, their closeness related to v can be unexpected high.

These two anomalous indication can be measured by Score 1 and Score 2 respectively, and the outlying score is the sum.

Definition 8 (Outlying Score). Given two snapshots G_{T-1} and G_T, Corenet$^{T-1}(v)$ and Corenet$^{T}(v)$ represent the Corenets of node v in G_{T-1} and $G_T$respectively. We denote the intersection of Corenet$^{T-1}(v)$ and Corenet$^{T}(v)$ except v as C_{old}, which is the set of old neighbors of node v. The elements of Corenet$^{T-1}(v)\backslash C_{old}$ are the neighbors removed from Corenet(v) at time T, denoted as $C_{removed}$. The elements of Corenet$^{T}(v)\backslash C_{old}$ are new neighbors of v, denoted as C_{new}. The outlying score of node v is defined as:

$$\begin{aligned} OutlyingScore(v) = & \sum_{v_i \in C_{old}} [closeness^{T-1}(v_i, v) - closeness^{T}(v_i, v)] \\ & + \sum_{v_r \in C_{removed}} closeness^{T-1}(v_r, v) \\ & + \sum_{\substack{v_j \in C_{new} \\ v_i \in C_{old}}} [(1 - \frac{w_{v_i v_j}}{w_{v_j}}) \times closeness^{T}(v_j, v)] \end{aligned} \tag{2}$$

Where $w_{v_i v_j}$ is the weight of edge between v_i and v_j, w_{v_j} is the sum of the weights of the edges connected to v_j.

The sum of former summation terms is Score 1, which measures outlying degree caused by situation (1). Similarly, the third summation term represents Score 2, which measures outlying degree caused by new neighbors in situation (2).

Algorithm. IcLEOD Algorithm (High level definition)

Input: Snapshots G_{T-1} and G_T, the number of closet neighbors related to the core k, the number of LEOutliers n;
Output: n LEOutliers

Step 1: Identify the time-varying components by comparing G_{T-1} and G_T;
Step 2: Determine incremental nodes collection IC based on time-varying components;
Step 3: For each node v in IC, compute Corenet$^{T-1}(v)$ and Corenet$^{T}(v)$;
Step 4: Compute outlying score for each object according to Eq.2;
Step 5: Select and output the objects with the first n-largest Outlying Score;

4 Experiments

In this section, we illustrate the general behavior of the proposed IcLEOD algorithm. Since there is no ground truth for outlier detection, we test the accuracy of our approach on multiple synthetic datasets with injected outliers. We also compare scalability performance of our approach with several baseline methods on synthetic datasets, and we present some meaningful cases obtained by our approach on real data set DBLP.

4.1 Baselines

We compare the proposed algorithm with the following three baseline methods:

- *CEOD*: This baseline is a community-based outlier detection method [3,6], which takes three necessary procedures to detect outliers evolving differently with their communities, including community discovery, community matching and outlier detection.
- *EGO*: In this approach, we regard the nodes in egonet are the closest neighbors of ego (node of interest), and we detect outliers by comparing the egonets at different timestamps.
- *SuperEGO*: This method is similar to *EGO* except that it considers neighbors within 2-hop as the ego's closest neighbors.

4.2 Data Description and Evaluation Measure

Synthetic Data Sets: We generate a variety of synthetic datasets, each of which consists of two snapshots.

First, we use the Butterfly generator [18] in order to generate datasets with normal nodes. The synthetic weighted graph follows WPL(weight power law) and SPL(snapshot power law), i.e., $W(t) = E(t)^{\alpha}$ and $W_n \propto d_n^{\beta}$. E(t), W(t) are the number of edges and total weight of a graph respectively at time t, W_n is the total weight of the edges attached to each node and d_n is the degree of the node. We set α and β to be 1.3 and 1.1 respectively.

Next, for each dataset, we inject outliers. We first set the percentage of outliers η, and inject $|\mathrm{V}|_{snapshot1} \times \eta$ outliers into datasets. $|\mathrm{V}|_{snapshot1}$ is the number of vertices in $\mathrm{Snapshot}_1$. Then we choose a random couple of objects e.g. v_1 and v_2, which exist in both $\mathrm{Snapshot}_1$ and $\mathrm{Snapshot}_2$. If v_1 and v_2 are far apart with common acquaintances few enough, we swap v_1 and v_2 in $\mathrm{Snapshot}_2$. Thus, we inject two outliers in the dataset. More detail information about synthetic datasets is shown in Table 2. Change ratio is the percentage of time-varying components.

Table 2. Summary of Synthetic Datesets

Dataset	$\|\mathrm{V}\|_{snapshot1}$	$\|\mathrm{E}\|_{snapshot1}$	$\|\mathrm{V}\|_{snapshot2}$	$\|\mathrm{E}\|_{snapshot2}$	Change Ratio
SYN1	1,000	6,520	1,054	7,093	9.42%
SYN2	5,000	19,762	5,109	20,251	3.56%
SYN2	10,000	29,415	10,184	30,019	2.01%

DBLP: We adopt DBLP as the real dataset (dblp.uni-trier.de/), which contains computer science scientific publications. In our representation, we consider a undirected co-authorship network. The weighted graph W is constructed by extracting author-paper information: each author is denoted as a node in W; journal and conference papers are represented as links that connect the authors

together; the edge weight is the number of joint publications by these two authors. We first removed the nodes with too low degree, than we extracted two co-authorship snapshots corresponding to the years 2001-2004 (13,511 authors) and 2005-2008 (14,270 authors).

We measured the performance of different algorithms using well-known metric F1 measure, which is defined as follows.

$$F1 = \frac{2 \times Recall \times Precision}{Recall + Precision}$$

Where recall is ratio of the number of relevant records retrieved to the total number of relevant records in the dataset; precision is ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved.

4.3 The Accuracy of IcLEOD Algorithm

We evaluate the accuracy of the proposed algorithms on the simulated datasets. The accuracy of the algorithms is measured by detecting the injected outliers as that of the groundtruths. We set the number of closet neighbors k to 30, 15, 10 for SYN1, SYN2 and SYN3, respectively. We vary the percentage of injected outliers η as 1%, 2% and 5%. In fairness to all algorithms, we perform 50 experiments for each parameter setting and report the average F1 of all algorithms. Table 3 illustrates the comparison results.

Table 3. The Accuracy Comparison on the Synthetic Datasets

Dataset	Outlier η	CEOD	EGO	SuperEGO	IcLEOD
SYN1	1%	0.1554	0.2012	0.2965	**0.8940**
	2%	0.2018	0.1912	0.2244	**0.7836**
	5%	0.1614	0.2845	0.3122	**0.9065**
SYN2	1%	0.0867	0.2150	0.4016	**0.7926**
	2%	0.1945	0.2631	0.4936	**0.8012**
	5%	0.2124	0.1983	0.6288	**0.9174**
SYN3	1%	0.2182	0.2064	0.5462	**0.7329**
	2%	0.3796	0.2042	0.4986	**0.7074**
	5%	0.1862	0.3216	0.6032	**0.8922**

As it can be observed from Table 3, the proposed algorithm (IcLEOD) outperforms the others in indicating outliers precisely for all the settings. It is clear that CEOD and EGO fail to find local evolutionary outliers. This is because the former identifies outliers form the view of entire community instead of the

local neighborhood substructure, and the latter only consider the neighbors with direct connectivity. The overall performance of SuperEGO is better than other baselines, but it significantly underperforms when the individual object's edge-weight distribution is clearly not uniform, like SYN1. This is due to SuperEGO ignores the edge-weight information. In contrast, the proposed algorithm detects outliers by considering both the local topology structure and the closeness transmission.

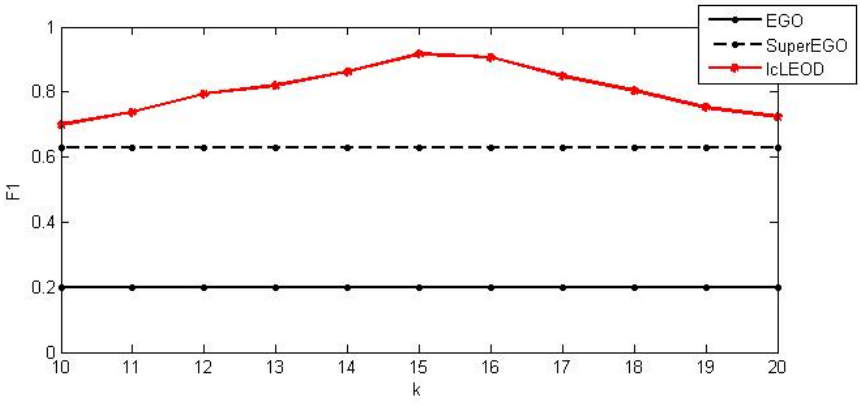

Fig. 3. Sensitivity

Figure 3 shows the sensitivity of the IcLEOD algorithm on parameter k. Two black lines represent the performance of baseline methods EGO and SuperEGO, respectively. We vary k from 10 to 20 for IcLEOD algorithm, as illustrated using the red line. The three algorithms are applied on the same data set, SYN2 and 5% outliers. Obviously, the proposed method is superior to two baseline methods, in spite of some changes caused by parameter variation.

4.4 The Scalability of IcLEOD Algorithm

To evaluate the scalability of IcLEOD, we conduct experiments on generated datasets as they vary the number of nodes. In Figure 4, the X-axis represents the number of nodes, whereas the Y-axis illustrates the computation time. We noticed that the processing time of the proposed approach is obviously lower than CEOD method. This is because the proposed approach only needs to calculate Corenets for nodes in IC (Def. 7), whereas CEOD method has to discover communities for entire network at each snapshot, even when there is no apparent change between two snapshots. Despite the EGO and SuperEGO approaches need less computation time, they have no specific procedure to determine the closeness neighborhood, which is likely to cause unfavorable results. The experiments demonstrate that there is a linear dependency of IcLEOD's processing

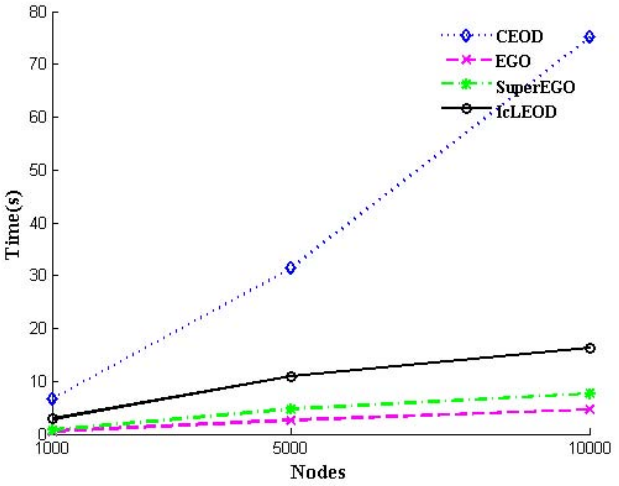

Fig. 4. Scalability Test of algorithms

time on the number of time-varying components in networks. Moreover, we can see that for the 10^4 network, the computation time is less than 20 seconds. This property means that the algorithm is practical in real applications.

4.5 Case Studies for Real Data Set

We will discuss an interesting outlier discovered by our algorithm on DBLP data set, which provides an intuitive perception about the effectiveness of our approach.

LEOutlier Case: [DBLP] Alexander Tuzhilin
We notice that Alexander Tuzhilin is a LEOutlier corresponding to DBLP 2001-2004 and DBLP 2005-2008. In DBLP 2001-2004 he was interested in Association Rules Analysis, and he shifted the focus of his research to Recommendation System in DBLP 2005-2008. We further noticed that his coauthors and the number of joint publications with these coauthors in two snapshots are very different. The principal members of his Corenets in two snapshots are listed as follows:

- Snapshot DBLP 2001-2004, Corenet_1(Alexander Tuzhilin): Tianyi Jiang, Hong Zhang, Balaji Padmanabhan, Gediminas Adomavicius etc.
- Snapshot DBLP 2005-2008, Corenet_2(Alexander Tuzhilin): Ada wai chee Fu, Cosimo Palmisano, Michele Gorgoglione, David jensen, Tianyi Jiang, Christos Faloutsos, Gueorgi Kossinets etc.

As the number of his publications increased, he established partnership with new researchers in recommendation system domain in the years 2005-2008 instead

of keeping or strengthening relationships with his coauthors in 2001-2004. The research field of most his former coauthors was still association rules analysis, still others turned research direction to other domains except recommendation system.

5 Conclusions

Since dynamic social networking applications are becoming increasingly popular, it is very important to detect anomalies in the form of unusual evolutionary behaviors. In this paper, we focus on outlier detection in *evolving weighted* graphs from a *local* perspective. We propose a novel outlier detection algorithm IcLEOD, to identify objects with anomalous evolutionary behavior particularly relative to their local neighborhoods. IcLEOD is an effective two-stage algorithm. In the first phase, we carefully design the local neighborhood subgraph named Corenet for individual object, which contains the node of interest and its closest neighbors in terms of associated structure and edge-weight information. To quantify how outlying an object is, we put forward a measurement in the second phase by analyzing and comparing the Corenets at different snapshots. IcLEOD algorithm is significant efficient for LEOutlier detection in gradually evolving networks, because it could avoid repeated calculations by incrementally analyzing the dynamic components. The experimental results on both real datasets and synthetic datasets clearly ascertain that the proposed algorithm is capable of identifying local evolutionary outliers accurately and effectively.

Future work could will concentrate on further refinement of IcLEOD algorithm for dealing with general evolving datasets with multiple snapshots efficiently.

Acknowledgment. This work was supported by the National High Technology Research and Development Program of China (Grant No. 2012AA011002), National Science and Technology Major Program (Grant No. 2010ZX01042-002-002-02, 2010ZX01042-001-003-05), National Science & Technology Pillar Program (Grant No. 2009BA H44B03), Natural Science Foundation of China 61073018, the Cultivation Fund of the Key Scientific and Technical Innovation Project, Ministry of Education of China (Grant No. 708001) and the Shenzhen-Hong Kong Innovation Cooperation Project (No. JSE201007160004A). We would like to thank anonymous reviewers for their helpful comments.

References

1. Barnett, V., Lewis, T.: Outliers in Statistical Data, 3rd edn. Wiley, New York (1994)
2. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. Technical Report, University of Minnesota (2007)
3. Gupta, M., Gao, J., Sun, Y., Han, J.: Integrating Community Matching and Outlier Detection for Mining Evolutionary Community Outliers. In: KDD (2012)

4. Ren, C., Lo, E., Kao, B., Zhu, X., Cheng, R.: On Querying Historical Evolving Graph Sequences. In: VLDB (2011)
5. Parthasarathy, S., Ruan, Y., Satuluri, V.: Community Discovery in Social Networks: Applications, Methods and Emerging Trends. In: Social Network Data Analytics. Springer, US (2011)
6. Gupta, M., Gao, J., Sun, Y., Han, J.: Community Trend Outlier Detection using Soft Temporal Pattern Mining. In: Flach, P.A., De Bie, T., Cristianini, N. (eds.) ECML PKDD 2012, Part II. LNCS, vol. 7524, pp. 692–708. Springer, Heidelberg (2012)
7. Gao, J., Liang, F., Fan, W., Wang, C., Sun, Y., Han, J.: On Community Outliers and their Efficient Detection in Information Networks. In: KDD (2010)
8. Aggarwal, C.C., Zhao, Y., Yu, P.S.: Outlier Detection in Graph Streams. In: ICDE (2011)
9. Flake, G., Lawrence, S., Giles, C.L.: In: SIGKDD (2000)
10. Bagrow, J.P., Bollt, E.M.: Phys. Rev. E (2005)
11. Saligrama, V., Zhao, M.: Local anomaly detection. In: AISTATS (2012)
12. Akoglu, L., McGlohon, M., Faloutsos, C.: Oddball: Spotting Anomalies in Weighted Graphs. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010, Part II. LNCS (LNAI), vol. 6119, pp. 410–421. Springer, Heidelberg (2010)
13. Ji, T., Gao, J., Yang, D.: A Scalable Algorithm for Detecting Community Outliers in Social Networks. In: Gao, H., Lim, L., Wang, W., Li, C., Chen, L. (eds.) WAIM 2012. LNCS, vol. 7418, pp. 434–445. Springer, Heidelberg (2012)
14. Aggarwal, C.C., Zhao, Y., Yu, P.S.: Outlier Detection in Graph Streams. In: ICDE (2011)
15. Breunig, M.M., Kriegel, H.-P., Ng, R.T., Sander, J.: LOF: Identifying Density-Based Local Outliers. In: SIGMOD (2000)
16. Aggarwal, C.C., Yu, P.S.: Outlier Detection with Uncertain Data. In: SDM (2008)
17. Knorr, E.M., Ng, R.T., Tucakov, V.: Distance-Based Outliers: Algorithms and Applications. The VLDB Journal 8, 237–253 (2000)
18. Mcglohon, M., Akoglu, L., Faloutsos, C.: Weighted graphs and disconnected components: patterns and a generator. In: KDD (2008)
19. Wasserman, S., Faust, K.: Social Network Analysis. Cambridge University Press (1994)
20. Neil, J.C., Fisk, M., Storlie, C., Brugh, A.: Graph-Based Network Anomaly Detection. In: JSM (2010)
21. Clauset, A., Moore, C., Newman, M.E.J.: Hierarchical structure and the prediction of missing links in networks. Nature 453, 98–101 (2008)