J Intell Inf Syst
DOI 10.1007/s10844-011-0183-2

Community-based anomaly detection in evolutionary networks

Zhengzhang Chen · William Hendrix · Nagiza F. Samatova

Received: 7 March 2011 / Revised: 2 August 2011 / Accepted: 6 October 2011

Abstract Networks of dynamic systems, including social networks, the World Wide Web, climate networks, and biological networks, can be highly clustered. Detecting clusters, or communities, in such dynamic networks is an emerging area of research; however, less work has been done in terms of detecting community-based anomalies. While there has been some previous work on detecting anomalies in graph-based data, none of these anomaly detection approaches have considered an important property of evolutionary networks—their community structure. In this work, we present an approach to uncover community-based anomalies in evolutionary networks characterized by overlapping communities. We develop a parameter-free and scalable algorithm using a proposed representative-based technique to detect all six possible types of community-based anomalies: grown, shrunken, merged, split, born, and vanished communities. We detail the underlying theory required to guarantee the correctness of the algorithm. We measure the performance of the community-based anomaly detection algorithm by comparison to a non–representative-based algorithm on synthetic networks, and our experiments on synthetic datasets show that our algorithm achieves a runtime speedup of 11–46 over the baseline algorithm. We have also applied our algorithm to two real-world evolutionary networks, Food Web and Enron Email. Significant and informative community-based anomaly dynamics have been detected in both cases.

This work was supported in part by the U.S. Department of Energy, Office of Science, the office of Advanced Scientific Computing Research (ASCR) and the Office of Biological and Environmental Research (BER) and the U.S. National Science Foundation (Expeditions in Computing). Oak Ridge National Laboratory is managed by UT-Battelle for the LLC U.S. D.O.E. under contract no. DEAC05-00OR22725.

Z. Chen · W. Hendrix · N. F. Samatova (✉)
Department of Computer Science, North Carolina State University, Raleigh, NC 27695, USA
e-mail: samatova@csc.ncsu.edu

Z. Chen · N. F. Samatova
Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA

Published online: 19 October 2011

Keywords Anomaly detection · Time-varying graphs · Evolutionary analysis · Community detection · Community-based anomaly

1 Introduction

As opposed to most research in anomaly detection, which is based on strings or attribute-value data as the medium, *graph-based* anomaly detection focuses on data that can be represented as a graph (Noble and Cook 2003). It has provided new approaches for handling data that can't be easily analyzed with traditional non–graph-based data mining approaches (Noble and Cook 2003) and has found applications in several domains. One of the most important of these areas is intrusion detection. GrIDS, a graph-based intrusion detection system, was developed by Staniford-chen et al. (1996). Padmanabh et al. (2007) proposed a random walk-based approach to detect outliers in Wireless Sensor Networks. Eberle and Holder (2006) focused on detecting anomalies in cargo shipments. Noble and Cook (2003) used anomaly detection techniques to discover incidents of credit card fraud (Eberle and Holder 2007).

Graph-based anomaly detection has been studied from two major perspectives: "*white crow*" and "*in-disguise*" anomalies. Intuitively, a "*white crow*" anomaly (also called an "outlier" in many papers) is an observation that deviates substantially from the other observations (Moonesinghe and Tan 2006), while an "*in-disguise*" anomaly is only a minor deviation from the normal pattern (Eberle and Holder 2007), as shown in Fig. 1. For example, if we are analyzing the voters list and we come across a person whose age is 322, then we can take that as a "*white crow*" anomaly, because the age of a voter will typically lie between 18 and 100. On the other hand, anyone who is attempting to commit credit card fraud would not want to be caught—a criminal would want his or her activities to look as normal as possible, which represents an "*in-disguise*" anomaly. Anomalies classified as "*white crow*" are usually detected as nodes, edges, or subgraphs, while "*in-disguise*" anomalies are now only identified through unusual patterns, including uncommon nodes or entity alterations. A summary of the various research directions in this area is shown in Fig. 2.

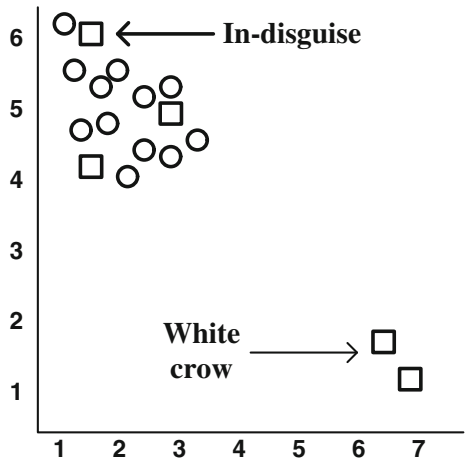

Fig. 1 "White crow" and "in-disguise" anomalies

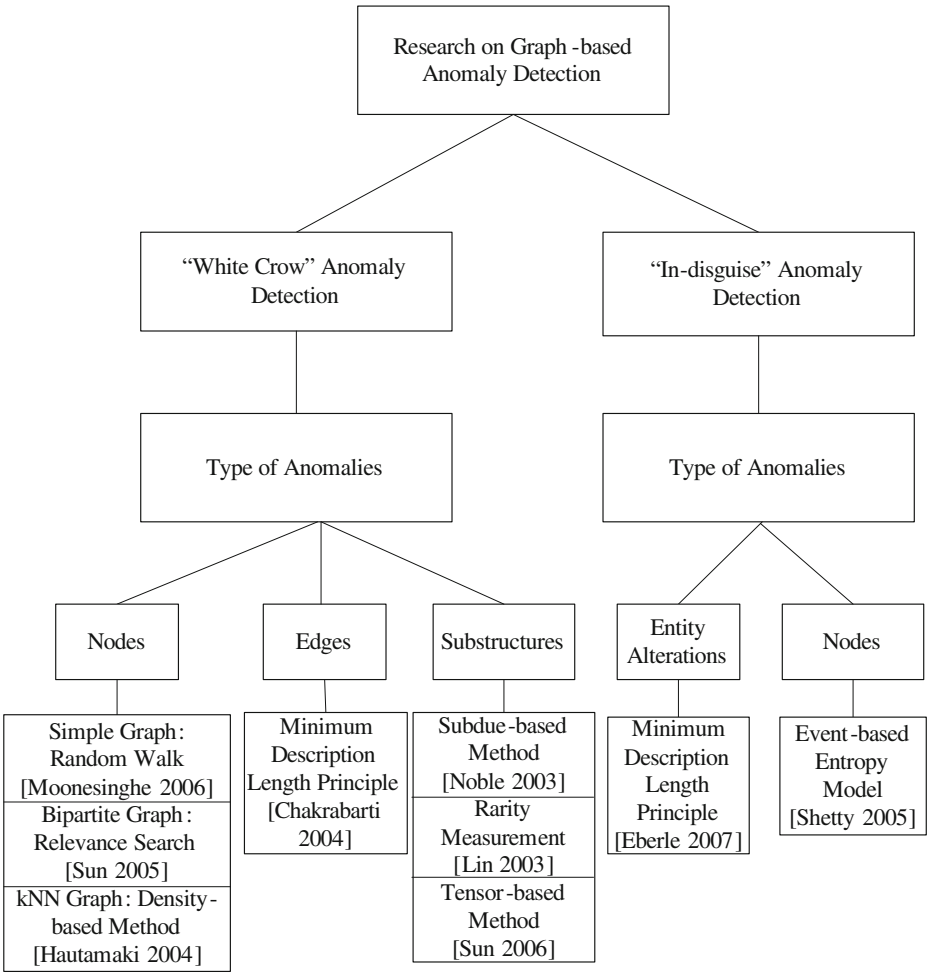

Fig. 2 A summary of the various research directions in graph-based anomaly detection

Research on *"white crow"* anomaly detection has traditionally focused on exploring three different types of anomalies. Aiming to identify *anomalous nodes*, Moonesinghe and Tan (2006) proposed a random walk–based approach that represents the dataset as a *weighted undirected* graph. Similarly, an algorithm based on random walks with restarts was used by Sun et al. (2005) for relevance search in an *unweighted bipartite* graph, in which vertices with low normality scores were treated as anomalies. Hautamäki et al. (2004) took a different approach and applied two density-based outlier detection methods to discover *novelty vertices* in a *k-nearest neighbour* graph. To identify *unusual edges*, Chakrabarti (2004) used the minimum description length principle to identify outlier edges, or the edges whose removal would best compress the graph. With the purpose of finding *abnormal patterns*, Noble and Cook (2003) used a variant of the minimum description length principle to deal with both anomalous substructures and anomalous subgraphs, based on their Subdue

system. In contrast to Noble and Cook (2003), Lin and Chalupsky (2003) applied rarity measurements to discover *unusually linked entities* within a labeled directed graph.

Unlike the previously mentioned *single graph* algorithms, Cheng et al. (2008) provided a robust algorithm for discovery of anomalies in noisy multivariate time series data. To deal with higher order data, Sun et al. (2006) introduced a tensor-based approach. Other related work on "*white crow*" anomaly detection can be found in Chan and Mahoney (2005), Sun et al. (2007), Keogh et al. (2005) and others.

"*In-disguise*" anomalies are more difficult to detect because they are highly hidden in the graph, and less work has been reported on detecting such anomalies. Eberle and Holder (2007) introduced three algorithms based on the minimum description length principle for the purpose of detecting three categories of anomalies that closely resemble normal behavior, including label modifications, vertex/edge insertions, and vertex/edge deletions. In addition, Shetty and Adibi (2005) exploited an event-based entropy model that combines information theory with statistical techniques to discover hidden prominent people in an Enron e-mail dataset. However, none of these work is focused on detecting "*in-disguise*" anomalies in *multiple graphs*. Meanwhile, neither "*white crow*" nor "*in-disguise*" anomaly detection approaches have considered one of the important properties of evolutionary networks: their community structure, which is sometimes referred to as clustering (Girvan and Newman 2002).

Networks of dynamic systems can be highly clustered (Watts and Strogatz 1998). A community, defined as a collection of individual objects that interact unusually frequently, is a very common substructure in many networks (Girvan and Newman 2002), including social networks, metabolic and protein interaction networks, financial market networks, and even climate networks. In social networks, a community is a real social grouping sharing the same interests or background (Girvan and Newman 2002). In biological networks, a community might represent a set of proteins that perform a distinct function together. Communities in financial market networks might denote groups of investors that own the same stocks, and communities in climate networks might indicate regions with a similar climate or climate indices.

Many algorithms have been developed for detecting community structures in static graphs. Girvan and Newman (2002) proposed a community discovery algorithm based on the iterative removal of edges with high *betweenness* scores. To reduce the computational cost of the betweenness-based algorithm, Clauset et al. (2004) proposed a modularity-based algorithm. In contrast, Palla et al. (2005) did not focus on detecting separate communities, but on finding overlapping communities. Defining communities as maximal cliques, Schmidt et al. (2009) proposed a parallel, scalable, and memory-efficient algorithm for their enumeration.

In addition, some work has been done on detecting *conserved* or *stable* communities in evolutionary networks. Hopcroft et al. (2004) proposed a method that utilizes a "nature community" to track stable clusters over time. A framework for identifying communities in dynamic social networks, proposed by Tantipathananandh et al. (2007), makes explicit use of temporal changes. Using the Clique Percolation Method to locate communities, Palla et al. (2007) defined auto-correlation and stationarity to characterize a community. From an application perspective, Steinhaeuser et al. (2009) provided a method to identify climate regions by detecting communities in time-varying climate networks.

Communities in real networks change over time, and being able to detect small community deviations can help us understand and exploit these networks more effectively. For example, in biological networks, a small variation in a gene-gene association community may represent an event, such as gene fusion (Snel et al. 2000), gene fission (Snel et al. 2000), gene gain (Chen et al. 1997), gene decay (Long et al. 2003), or gene duplication (Zhang 2003), that would change the properties of the gene products (e.g., proteins) and, consequently, affect the phenotype of the organism. Interesting community deviation patterns in Food Web and social networks are discussed in Section 3.

Thus, in contrast to the previous work on identifying communities or tracking *conserved* communities, we focus on detecting *community-based anomalies*, a new type of "*in-disguise*" anomalies, in dynamic networks. Specifically, our work proposes the novel problem of detecting these "*in-disguise*" anomalies across *multiple* dynamically evolving graphs, or *evolutionary networks*, for short. Our approach follows from the need to address the following four challenges:

- How do we define community-based anomalies, and how many types of community-based anomalies are possible in evolutionary networks? From an anomaly detection perspective, we are interested in dynamic, abnormal communities that would reveal novel properties of the network, as opposed to conserved communities or communities at a single snapshot. For example, is there a community at snapshot t that splits into smaller communities or merges with others at snapshot $t + \Delta t$? Does any community at snapshot t disappear at snapshot $t + \Delta t$, or does any new community appear at snapshot $t + \Delta t$? Do the sizes of the communities change over time?
- Most real networks are dynamic and characterized by overlapping communities (Palla et al. 2005). Detecting dynamic communities from networks characterized by overlapping communities is more challenging than discovering communities in static networks.
- How do we detect community-based anomalies across multiple dynamically evolving networks? As we mentioned earlier, real-world networks change over time, requiring us to adopt evolutionary analysis techniques to detect such anomalies.
- Since there may be hundreds or even thousands of communities in each real-world network, how to scale a community-based anomaly detection algorithm to large graphs?

In this paper, we propose solutions to all four of these problems. Our algorithm is based on the proposed notion of *graph representatives* and *community representatives*. *Graph representatives* helps us reduce the expensive computational cost of enumerating communities, which we model as maximal cliques, whereas *community representatives* is utilized to identify community-based anomalies.

The contributions of our work are:

1. We identify a new type of anomaly—the *community-based* anomaly in complex evolutionary networks.
2. Our work tackles the unexplored question of detecting "*in-disguise*" anomalies across *multiple* graphs.

3. We prove that there are only six possible types of community-based anomalies in dynamic simple undirected graphs.
4. We develop a community-based anomaly detection algorithm based on *graph representatives* and *community representatives*.
5. We evaluate our method on real datasets to confirm its applicability in practice.

The rest of the paper is organized as follows: Section 2 introduces some necessary definitions and formally defines the problem. In Section 3, we show application of community-based anomaly detection to two real-world dynamic networks, Food Web and Enron Email. Section 4 presents the community-based anomaly detection algorithm. In Section 5, we evaluate the algorithm with synthetic data. Finally, Section 6 concludes the paper.

2 Problem statement

In this paper, the ultimate goal is to find community-based anomalies in dynamic graphs, and our algorithm is based on *graph representatives* and *community representatives*. Thus, the following terms and problems need to be addressed. The symbols used in the paper are listed in Table 1.

Problem 1 (Community-based anomaly detection) Given a time-varying sequence of undirected simple graphs $\mathcal{G} = \{G_1, G_2, G_3, ...\}$, where the nodes in each graph can belong to different communities, detect the community-based anomalies between consecutive graphs, including grown, shrunken, merged, split, born, and vanished communities (see Definition 7).

Definition 1 (Community) Communities are the maximal cliques in a graph.

There is no formal definition for the community structure in a network (Girvan and Newman 2002). The simplest and the most conservative definition of a commu-

Table 1 Symbol table

Symbol	Description
G_i	A simple undirected labeled graph
$\mathcal{G}$	A sequence of graphs
C_t^i	The community of index i in graph G_t
$Rep(G_i)$	The representative node set of graph G_i
$C_t^i \rightarrow C_{t+1}^j$	C_t^i is a predecessor of C_{t+1}^j, or C_{t+1}^j is a successor of C_t^i
$V(C_t^i)$	The node set of community C_t^i
$SV(G_i)$	Common nodes between graphs G_i and G_{i+1}
$\|C\|$	The size of community C
T	The number of timestamps in the sequence
$\|V(C)\|$	The number of vertices in community C
v_j	A vertex j in a graph
CG_i	The list of communities in G_i
$VC_i^{v_j}$	The list of communities that contain node v_j in graph G_i
$Checked(G_i)$	The list of nodes in G_i that have been checked
$\emptyset$	The empty set

nity is a *clique*, a set of vertices that are pairwise adjacent to one another. Another definition used by Girvan and Newman (2002) is a dense subgraph, a group of vertices within which the connections are denser than between different groups (Girvan and Newman 2002).

As our goal is to detect abnormal, changing communities, we propose to use the more specific community definition, namely clique. From an application perspective, we could lose important information if we shifted to dense subgraphs as communities. For example, consider protein functional modules (biological communities) in protein-protein interaction networks. Across different organisms, such evolutionary networks might have undergone small changes, or perturbations, due to evolutionary events such as gene fusion, gene fission, gene gain, gene decay, or gene duplication. Relatively small perturbations to the network structure due to the genotype variation may induce phenotype variations, such as organism's capability to produce hydrogen or ethanol, to resist high temperature, to fix nitrogen, etc. Since network perturbations could be infinitesimal, considering communities as dense subgraphs with respect to some density parameter, may arguably be insufficient for capturing such fine-grain changes to the network structure. Therefore, we propose to use the simplest, the most stringent, and parameter-free definition of a community—a clique. We use the maximal clique, i.e., a clique that can't be extended by adding any more vertices in order to decrease the space of putative anomaly-based communities to evaluate and thus to reduce the overall computational cost.

Definition 2 (Community size) The community size, $|C|$, is the number of vertices in the community, so $|C| = |V(C)|$.

Definition 3 (Graph representative) Representatives of graph G_i are the nodes that also appear in G_{i-1}, G_{i+1}, or both. Thus, $Rep(G_i) = \{v_i \mid v_i \in V(G_i) \cap (V(G_{i-1}) \cup V(G_{i+1}))\}$.

Nodes that only appear in one graph are called graph-specific nodes or vertices.

Definition 4 (Graph-specific community) A graph-specific community is a community that does not contain any graph representative.

Since our goal is to detect community-based anomalies, we do not try to discover graph-specific communities. Thus, by using graph representatives as seeds, we do not need to enumerate all communities in the graphs, only those communities that contain graph representatives, and thus potentially reducing computational time (see Section 4 for details).

Definition 5 (Community predecessor and successor) If community C_t^i at snapshot t is a subset or superset of community C_{t+1}^j at snapshot $t+1$, then the community C_t^i is a predecessor of C_{t+1}^j, and C_{t+1}^j is a successor of C_t^i. This relationship is denoted by $C_t^i \rightarrow C_{t+1}^j$.

Definition 6 (Community representative) A community representative of C_t^i is a node in C_t^i that has the minimum number of appearances in other communities of the

same graph. If there is more than one node that satisfies this condition, we choose one at random.

The rationale for our definition of a community representative follows from the observation that the community C_t^i can be represented by a node that only appears in community C_t^i. However, since the communities in our networks may be highly overlapping, we cannot guarantee that such a node exists, so we look for a node in C_t^i that has the minimum number of appearances in other communities to use as its representative. In this way, we limit the nodes that belong to more than one community from being a community representative, which helps to establish the relationships between the communities (see Section 4 for details).

Definition 7 (Community-based anomalies) In contrast to Palla et al. (2007), which focuses on the stability/stationarity of the communities, our goal is to detect community-based anomalies. As there are six basic events that may occur to a community (Palla et al. 2007), we can define six possible types of community-based anomalies in evolutionary networks (see Fig. 3).

1. *Grown community*
   In real-world networks, some "big" communities, like community 2, can be grown from previous "smaller" communities by adding some new members. These "big" communities are called grown communities.

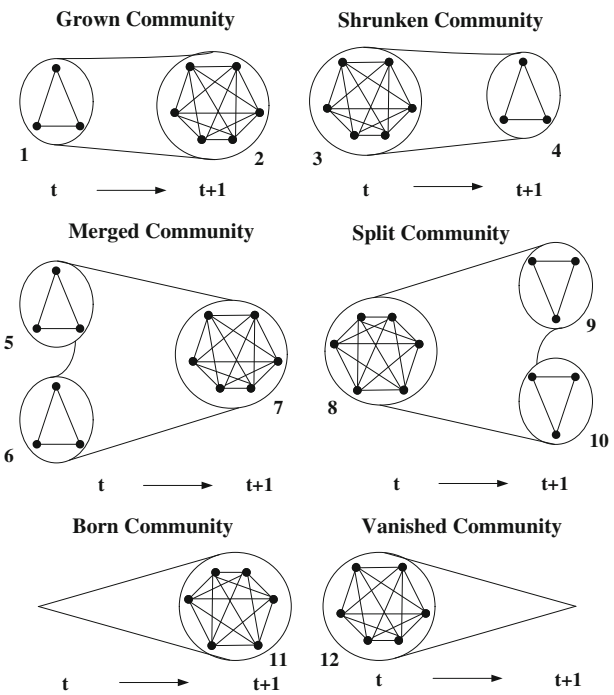

Fig. 3 Possible types of community-based anomalies in evolutionary networks

2. *Shrunken community*
   On the other hand, shrunken communities, like community 4, are communities caused by previous "bigger" communities losing some members.
3. *Merged community*
   In addition, two or more "small" communities at snapshot t often join together to form one merged community, like community 7, at snapshot $t + 1$.
4. *Split community*
   Meanwhile, a split community at snapshot t, like community 8, may break up into multiple communities at snapshot $t + 1$.
5. *Born community*
   What's more, some "new" communities, like community 11, may appear in some snapshots, but born communities should contain at least one graph representative in order to avoid considering graph-specific communities as anomalies.
6. *Vanished community*
   Alternatively, some "old" communities, like community 12, may disappear. Similar to born communities, vanished communities should contain at least one graph representative to exclude graph-specific communities.

Evolutionary network conservation, which is often manifested with stable communities that do not change over time, is a well-recognized property of many real-world complex dynamic networks. For example, in climate networks, such communities may correspond to well-known climate indicies. Likewise, in biological networks, such stable communities may correspond to protein complexes, such as ATP synthase or ribosomal machinery, and metabolic pathways, such as TCA cycle. In contrast to stable communities, an anomalous community is often highly hidden among an enormous number of stable communities in evolutionary networks. In real-world networks, like social networks, a majority of people's friendship communities tend to be stable despite frequently occuring changes in individuals' activities and communication patterns (Palla et al. 2007).

It is often the case, especially if Δt is small, that only very few communities might slighly change due to some anomalous events. For example, resignation of the CEO in a company may induce changes to community composition, if community membership is defined by email communication traffic between a sender and a receiver. Likewise, in climate networks, the seasons of unusually high hurricane activity are likely induced by changes in climate communities found in the climate networks for the seasons with low hurricane activity. Thus, rare and anomolous events are likely caused by or induce structural changes in the communities, and result in the appearance of anomalous communities. Such anomalous communities often overlap with other "normal" (or stable) communities, which makes it even more difficult to distinguish between normal and anomolous communities. Thus, community-based anomaly can be seen as a new type of "in-disguise" anomaly.

3 Application of community-based anomaly detection to real-world dynamic networks

In addition to the more controlled experiments using synthetic graph data sets, as described in Section 5, we applied our algorithm to two real-world dynamic networks,

Table 2 Food Web communities

Season	Number of communities	Abnormal communities
Spring	15	None
Summer	15	Four grown communities, one born community, four communities will split in fall, one vanished community, and one merged community
Fall	19	Two shrunken communities and four vanished communities (will disappear in winter)
Winter	9	Four merged communities

Food Web and Enron Email. In this section, we consider only communities of size three or more.

Food Web dataset The Food Web dataset, which was originally compiled by Baird and Ulanowicz (1989), consists of marine organisms living in the Chesapeake Bay, containing 33 vertices that represent the ecosystem's most prominent taxa. Edges

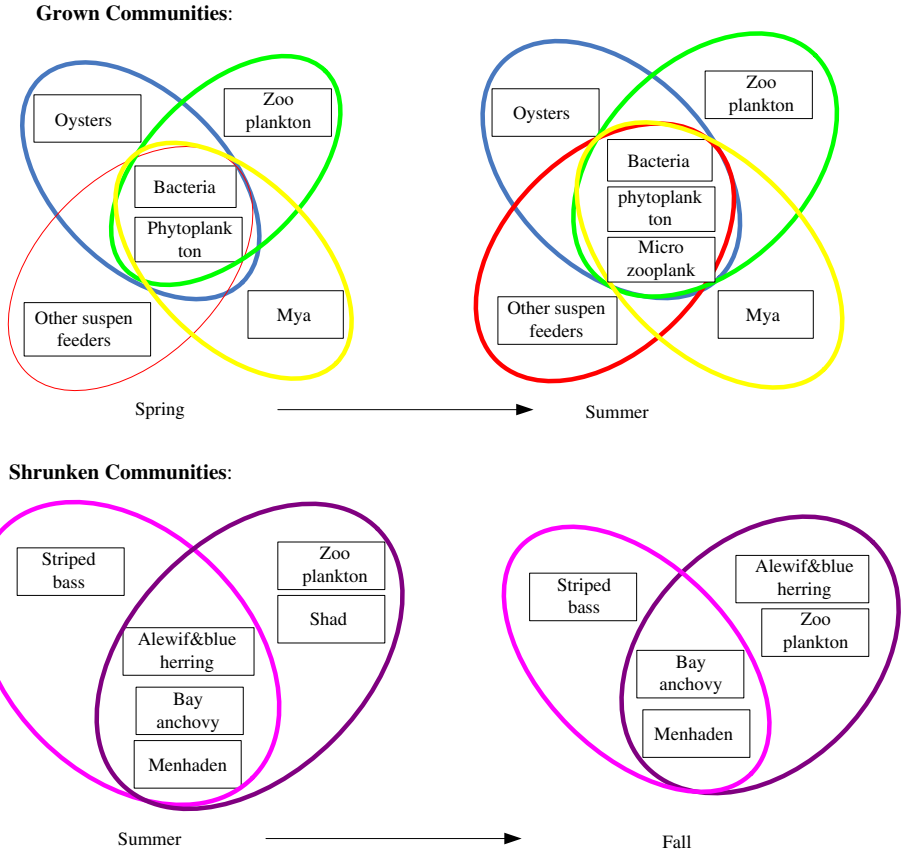

Fig. 4 Example of a grown community and a shrunken community in Food Web

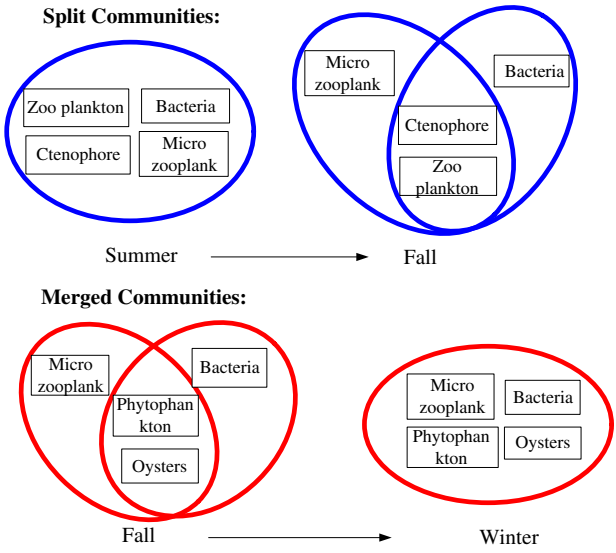

Fig. 5 Example of a split community and a merged community in Food Web

between taxa denote trophic relationships—one taxon feeding on another. Here, we ignore directionality and consider the network as an undirected graph. Girvan and Newman (2002) has used this dataset as a static graph to detect the communities, while we construct the networks on a seasonal basis from spring to winter to discover community-based anomalies in the dynamic network.

By applying our algorithm to the Food Web networks, we find instances of all six types of community-based anomalies (see Table 2). Summer is the most active community changing season: four communities grow because microzooplankton, which cannot be found in Spring, become involved in the energy flow network. Four communities split because bacteria do not feed on microzooplankton in Fall. The disappearance of sea nettle in the Fall results in a vanished community (zooplankton, ctenophore, and sea nettle). This community was a born community in Summer, indicating that it is unstable. Due to a lack of food in Winter, four communities merge in order to benefit from more members with food energy. Typical community-based anomaly examples discovered in Food Web are shown in Figs. 4, 5 and 6. Note that

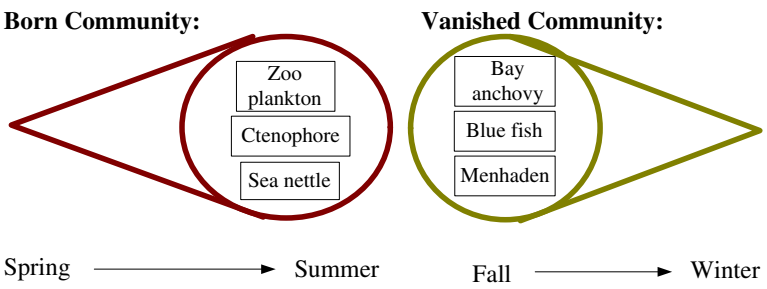

Fig. 6 Example of a born community and a vanished community in Food Web

Table 3 Enron email dataset properties

Month	Number of edges	Number of communities
January	126	21
February	190	56
March	199	54
April	240	66
May	273	90
June	218	49
July	240	68
August	371	120
September	343	110
October	531	196
November	438	143
December	290	93

the different circles represent different communities at the same time stamp—we can see that Food Web is characterized by overlapping communities.

ENRON dataset This data set consists of approximately 1.5 million email communications sent or received by employees in Enron, Inc. It is much more complex than the Food Web dataset. We take a subset containing only messages between Enron employees from January to December of 2001 and construct sender-to-recipient undirected graphs on a monthly basis. The graphs have 151 nodes (Enron employees), with low edge density and short average distance between vertices, which shows a "small-world" effect and indicates that the graphs have community structure. The properties of each graph are shown in Table 3.

The community-based anomalies in each month discovered by our algorithm are shown in Table 4. We can see that there are more abnormal communities in October than in any other month. The most likely trigger of this event is the fact that Enron announced a third quarter loss of $618 million on October 16 of 2001, which is also thought to be the trigger of the Enron scandal.

In order to see the details of the community-based anomalies in October, let us consider one of the most important nodes—Louise Kitchen, the former President of Enron. There are 20 abnormal communities containing Louise Kitchen in October:

Table 4 Community–based anomalies in enron email dataset

Month	Grown	Shrunken	Merged	Split	Born	Vanished
January	0	0	0	0	0	14
February	3	1	0	1	48	32
March	3	2	4	1	33	39
April	6	1	0	2	42	43
May	3	4	0	1	75	76
June	3	3	0	1	34	38
July	1	0	2	0	58	54
August	9	4	2	2	97	89
September	8	9	3	1	79	56
October	30	5	10	8	136	160
November	7	13	0	9	102	97
December	2	17	2	0	54	14

16 born communities, 4 grown communities, 1 split community, and 1 shrunken community. From Fig. 7, we can see that some employees like Sally Beck, Chief Operating Officer, joined the senior management communication groups, probably to discuss the serious issues or suggest strategies, while only one person—Phillip

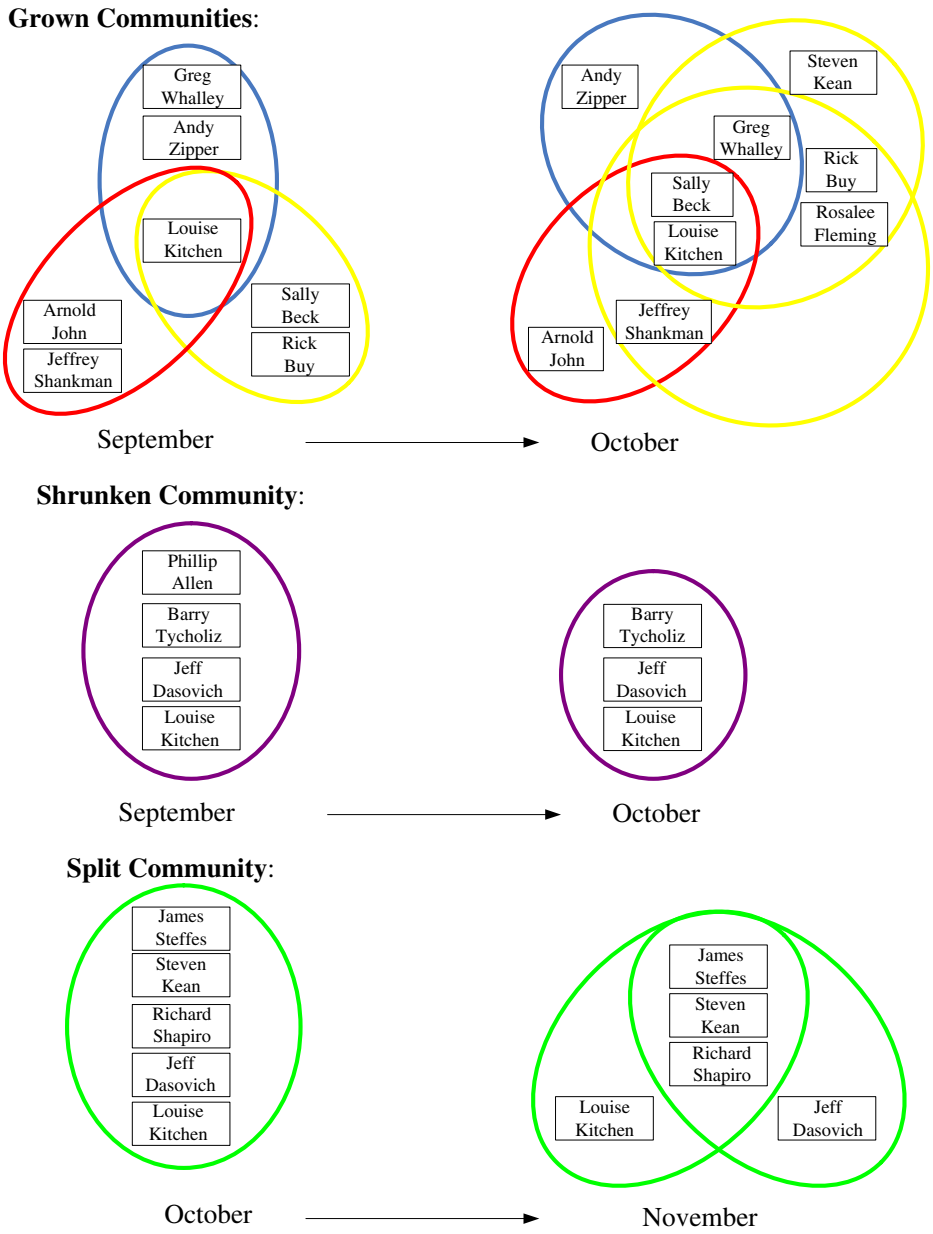

Fig. 7 Abnormal communities containing Louise kitchen in October

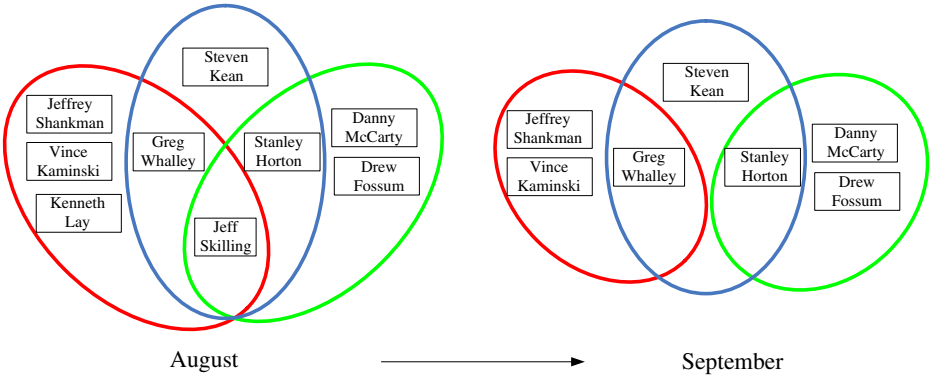

Fig. 8 Shrunken communities due to Jeff Skillng resigning as CEO in August

Allen—left the groups. Confusion among the Enron employees may be why Louise Kitchen's communication groups grew rather than shrank during the turbulent times. As a second example, take Jeff Skilling, the former CEO of Enron. There are 19 email communities contain Jeff Skilling in August, but among these, three communities shrank in September (see Fig. 8), while the other 16 communities disappeared after Jeff Skilling resigned as CEO in August, perhaps because many employees quit or joined other work groups after Skilling's resignation.

4 Community-based anomaly detection algorithm

In this section, we discuss the proposed algorithm for solving the problem presented above. We prove some necessary lemmas and theorems in Section 4.1. Then, based on the abnormal community decision rules described in Section 4.2, we illustrate how to detect community-based anomalies in Section 4.3.

4.1 Lemmas and theorems

We present the following theorems and lemmas to provide a sound theoretical basis for our community-based anomaly detection. Proofs for these appear in Appendix.

Lemma 1 *If community C_t^i has more than one predecessor (or successor), then the sizes of its predecessors (or successors) are either all larger than $|C_t^i|$ or all smaller than $|C_t^i|$.*

Similarly, if a community C_t^i has more than one successor, then the sizes of its successors are either all larger than $|C_t^i|$ or all smaller than $|C_t^i|$. This lemma is used to prove the following completeness result:

Theorem 1 *Let G_t and G_{t+1} both be simple, undirected graphs, where communities are defined as maximal cliques. If G_{t+1} is the perturbed graph formed by either adding edges/nodes to or removing edges/nodes from the baseline graph G_t, then there are only six possible types of community-based anomalies between G_t and G_{t+1}: grown*

communities, shrunken communities, merged communities, split communities, born communities, and vanished communities, as defined in Definition 7.

Lastly, we present a theorem that will allow us to reduce the computational complexity of identifying the community-based anomalies.

Theorem 2 *If community C_t^i is represented by vertex $v_i \in C_t^i$, and community C_{t+1}^j is represented by vertex $v_j \in C_{t+1}^j$, where $C_t^i \to C_{t+1}^j$, then $v_i \in C_{t+1}^j$ or $v_j \in C_t^i$.*

From Theorem 2, if there is more than one node in community C_t^m and C_{t+1}^n that satisfies the condition of community representative, then we can randomly choose v_m to represent C_t^m and v_n to represent C_{t+1}^n to help check the relationship between C_t^m and C_{t+1}^n as follows:

1. If $v_m \in C_{t+1}^n$, then C_{t+1}^n can be detected as a potential successor to C_t^i. By Definition 5, the relationship $C_t^m \to C_{t+1}^n$ would be established if C_{t+1}^n also satisfies $C_t^m \subseteq C_{t+1}^n$ or $C_{t+1}^n \subseteq C_t^m$.
2. If $v_m \notin C_{t+1}^n$, then by Theorem 2, $v_n \in C_t^m$ if $C_t^m \to C_{t+1}^n$. In this case, we can detect the relationship $C_t^m \to C_{t+1}^n$ through v_n and check whether $C_t^m \supset C_{t+1}^n$.

Thus, random selection of the community representative will not affect our detection results.

4.2 Abnormal community decision rules

Based on our result from Theorem 1, we can identify community-based anomalies using the following rules:

1. If community C_t^i has only one predecessor C_{t-1}^j:
   (a) If the size of the predecessor is smaller than $|C_t^i|$, then C_t^i is a grown community.
   (b) If the size of the predecessor is larger than $|C_t^i|$ and C_t^i is the only successor of C_{t-1}^j, then C_t^i is a shrunken community.
   (c) If the size of the predecessor is larger than $|C_t^i|$ and C_t^i is not the only successor of C_{t-1}^j, then C_t^i is a product of the split community C_{t-1}^j.
2. If community C_t^i has more than one predecessor:
   (a) If the sizes of the predecessors are all smaller than $|C_t^i|$, then C_t^i is a merged community.
   (b) If the sizes of the predecessors are all larger than $|C_t^i|$ and C_t^i is the only successor of one of its predecessors, then C_t^i is a shrunken community.
   (c) If the sizes of the predecessors are all larger than $|C_t^i|$ and C_t^i is not the only successor of one of its predecessors, then that community is a split community and C_t^i is one of its products.
3. If community C_t^i has no predecessor, then C_t^i is a born community.
4. If community C_t^i has no successor, then C_t^i is a vanished community.

4.3 Algorithm description

In this section, we describe our method for detecting and tracking anomalous communities based on the proposed notion of *graph representatives* and *community representatives*.

To the best of our knowledge, the proposed problem of *detecting and tracking community-based anomalies in evolutionary networks* has not been addressed in literature. Thus, for comparison purposes, we first briefly describe a brute-force solution that does not use graph representatives and community representatives. Then, we provide details on how *graph representatives* can help reduce the expensive computational cost caused by community enumeration, and how *community representatives* can be utilized to effectively identify community-based anomalies.

Non-representative-based method A brute-force solution that does not use graph or community representatives is to first enumerate all communities in each graph, and then compare all possible pairs of communities belonging to consecutive timestamps. For example, to find the successors of community A in Fig. 9, we need to compare community A with communities D, E, F, and K at snapshot $t+1$; that is, we compare community A with all communities at snapshot $t+1$, although only community F is the successor of A. This two-stage approach is infeasible and impractical, because of a possibly enormous number of communities to search. Among those, there are many redundant communities (e.g., graph-specific communities (see Definition 4)) in each graph, and it does not make much sense to compare pairs of communities that contain no common members or few members.

Representative-based method To reduce the computational cost, we designed an algorithm based on the graph representatives and community representatives (see Definition 3 and 6 in Section 2). The workflow of the algorithm is shown in Fig. 10. For each graph, we first find graph representatives (see Step 1 in Fig. 10) and enumerate the communities that are seeded by the graph representatives to avoid generating graph-specific communities (see Step 2 in Fig. 10). We call these communities seed-communities. In every seed-community, we select only one node as a community representative (see Step 3 in Fig. 10) and use community representatives to establish

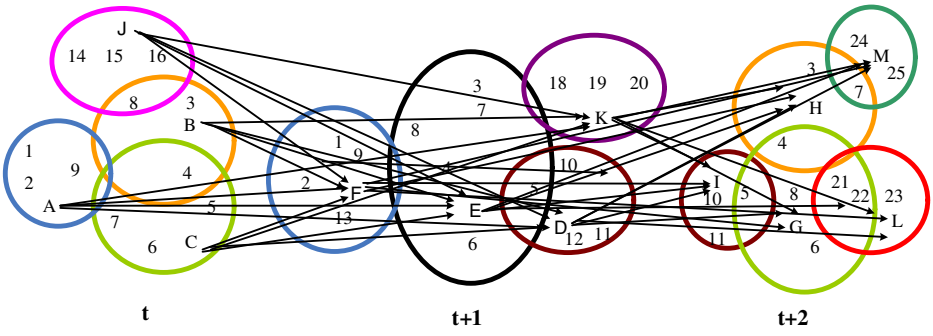

Fig. 9 Example for tracking community-based anomalies using the non–representative-based method

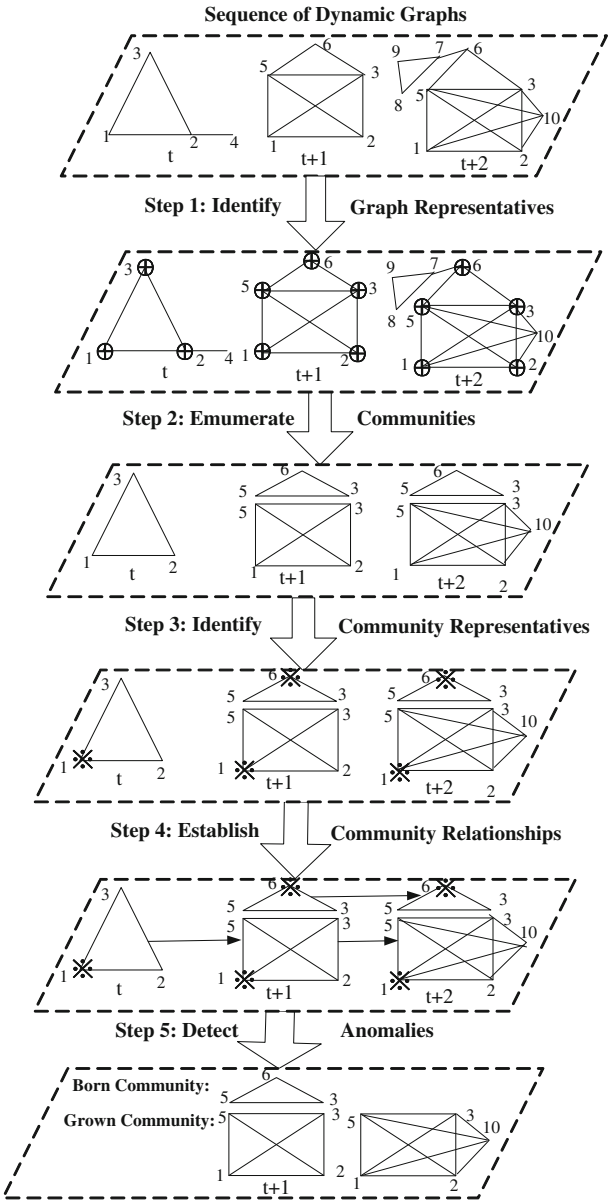

Fig. 10 Workflow of the community-based anomaly detection algorithm

predecessor–successor relationships between a pair of seed-communities from two consecutive graphs (see Step 4 in Fig. 10). Once all the predecessors and successors of the community C_t^i have been found, we apply the abnormal community decision rules in Section 4.2 to determine the type of anomaly present, if any (see Step 5 in Fig. 10).

Let us apply the representative-based algorithm to the same example in Fig. 9. Instead of enumerating all communities, the algorithm first identifies the set of

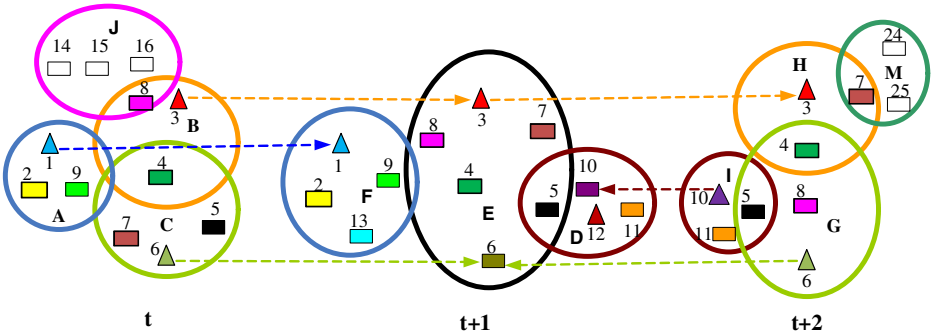

Fig. 11 Example for tracking community-based anomalies using the representative-based method. *Triangles*: community representatives; *Filled shapes*: graph representatives; *Empty shapes*: graph-specific vertices; *Circles*: communities; *Dashed lines*: predecessor–successor community relationships

graph representatives, which are the filled triangle or rectangle nodes highlighted in Fig. 11. Using graph representatives as seeds to generate communities, graph-specific communities (see Definition 4), like communities K and L in Fig. 9, now disappear (see Fig. 11). This strategy could possibly save a lot of computational cost on community enumeration. Once, we generate the seed-communities in each graph, the algorithm searches for community representatives (triangular nodes in Fig. 11) by selecting the vertices that appear in the fewest number of communities.

Taking advantage of community representatives, the algorithm can establish the predecessor–successor community relationships much more efficiently. Let us take community A at timestamp t, for example. To find the successor(s) of community A, the algorithm first finds all the communities that contain the community representative of A at timestamp $t+1$. In this case, only community F contains the community representative. Then, the algorithm checks whether community F is a subset or superset of A (see Definition 5). Only if one of these two conditions holds true does the algorithm establish the predecessor–successor relationship between A and F. When there are only grown, merged, born, or vanished community anomalies, the algorithm does not need to consider commmunities earlier in the sequence of graphs. For example, community A grows into community F, communities B and C merge into E, community D emerges, and community F disappears. However, in cases of shrunken or split communities, the algorithm may need to "backtrack" by using the representative of community C_t^i to look for its predecessors at timestamp $t-1$. For example, community D shrinks into I with the disappearance of representative 12 at timestamp $t+2$, and community E splits into H and G with representative $3 \in H$ but $3 \notin G$. To detect these anomalies, the algorithm needs to connect communities G and I at timestamp $t+2$ to communities E and D, respectively, at timestamp $t+1$ by "backtracking" the community representatives of G and I (6 and 10).

From Fig. 11, we can also see that if there is more than one node in the same community with the minimum number of appearances in other communities, then randomly choosing one node as a community representative would not affect the

detection results. For example, if we choose 9 instead of 1 as representative of community A or 8 instead of 3 as representative of B, we will still identify F as successor of A and E as the successor of B, since all nodes in the predecessors of the grown (or merged) community will also appear in the grown (or merged) community. Namely, if the representative of C_t^i appears in the successors of C_t^i, then we will find all such successors for C_t^i, because we check all communities that contain the representative of C_t^i at timestamp $t+1$. Even if the representative of C_t^i disappears in some successors of C_t^i, we can still establish the relationship between C_t^i and its successors by "backtracking" from the representatives of its successors, like the example of communities G, I, E, and D shown previously.

Once the community relationship is established, the algorithm uses the decision rules (see Section 4.2) to determine whether a community is an anomaly, based on the numbers and sizes of its predecessors and successors:

- A grown community, like community F in Fig. 11, can be detected by comparing the communities at the prior timestamp that contain the community representative 1 of F (community A, in this case). Because community F is larger than its predecessor A and has no other predecessors, based on decision rule 1a, community F is a grown community.
- A shrunken community, like community J in Fig. 11, can be detected by decision rule 1b, since it has only one larger predecessor.
- A community, such as community E that has two smaller successors–community H and G is identified as a split community (see decision rule 1c).
- A community, such as community E, is also detected as a merged community by the algorithm, because it has two smaller predecessors, community B and C, at timestamp t (see decision rule 2a).
- A community, like community M that has no predecessor (see decision rule 3), is detected as a born community.
- A community, like community J that has no successor (see decision rule 4), is identified as a vanished community.

We give a pseudocode description for our representative-based community anomaly detection algorithm in Algorithm 1. The input to the Algorithm 1 is a sequence of undirected graphs. Lines 1–12 are concerned with finding the graph representatives for each graph in the sequence and enumerating all the communities in each graph using the graph representatives as seeds. In lines 13–20, the algorithm calculates the number of times each node in each seedcommunity appears in each graph. It chooses one node with the fewest occurence in each community as the community representative (line 22). In line 23 through line 28, the algorithm establishes predecessor–successor community relationships. Since some community representatives may disappear in the successors of the community, lines 29 through 35 backtrack to establish community relationships between communities in the preceding timestep. This way, the algorithm can establish all community relationships. Finally, lines 37 through 41 apply the abnormal community decision rules to these relationships to identify the community-based anomalies. Thus, if any community in each graph belongs to one of six possible types of community-based anomalies, the algorithm will detect this anomaly and return its anomaly type.

5 Effectiveness of representative-based methodology

In this section, we evaluate the community-based anomaly detection algorithm on synthetic graph datasets to have a more controlled settings for assessing alogirthmic performance. These experiments complement the discoveries and insights offered by our algorithm when applied to real-world network data, Food Web and Enron Email datasets, as described in Section 3. Specifically, we focus on answering the following two questions:

1. What is the performance of the community-based anomaly detection algorithm using our representative-based technique?
2. Is our algorithm scalable to large graphs?

We study the performance of the community-based anomaly detection algorithm relative to the non–representative-based algorithm on synthetic networks of increasing size. Our experiments were conducted on a PC with an Intel Core 2 Duo CPU (2.1GHz) and 4GB of RAM. Our algorithm was implemented in the C programming language, and is available upon request. We measure the improvement in the runtime of our algorithm versus the non-representative-based algorithm in terms of speedup, which we calculate by dividing the runtime of non–representative-based algorithm to the runtime of our algorithm.

In this experiment, we study the effectiveness of the proposed representative-based technique. All the graphs in the synthetic datasets are generated by GT-graph (Bader and Madduri 2006) and follow the Recursive Matrix Graph model (R-MAT) (Chakrabarti et al. 2004) so that they have a small-world nature. The parameters for the synthetic graphs, which appear in Table 5, are defined as follows: $|V|$ is the number of vertices in a graph, Num_{gv} is the number of graph-specific vertices in a graph, and E_i is the number of edges in a graph G_i. On all graphs, we use default values of 0.45, 0.15, 0.15 and 0.25 for the R-MAT parameters a, b, c, d, with $a : b$ and $a : c$ ratios of 3:1, as in many real world graphs (Chakrabarti et al. 2004). After graph enumeration, we use a program to re-label some of the vertices according to the parameter Num_{gv}, so that we can have some graph-specific vertices in each graph when we build the sequence of graphs. For example, in the dataset *syn_500*, we can relabel the vertices $v_j \in [451, 500]$ in graph G_i as $v_j + 50 * (i - 1)$. Other graph-specific vertices in other datasets can be similarly re-labeled.

In the first experiment, we try to test the collective effectiveness of graph representatives and community representatives in our algorithm. We measure the entire runtime of the representative-based algorithm and the non–representative-based algorithm in each of the synthetic datasets. The result of the experiments

Table 5 Summary of synthetic datasets

Dataset	$\|V\|$	Num_{gv}	E_1	E_2	E_3	E_4	E_5
syn_500	500	50	8,000	11,000	9,000	12,000	10,000
syn_1000_1	1,000	100	400,000	550,000	45,0000	60,0000	50,0000
syn_1000_2	1,000	200					
syn_1000_3	1,000	300					
syn_1500	1,500	150	64,0000	880,000	720,000	96,0000	800,000
syn_2000	2,000	200	80,0000	1,100,000	900,000	1,200,000	1,000,000
syn_3000	3,000	300	160,0000	2,200,000	1,800,000	2,400,000	200,0000

Algorithm 1: Community-based anomaly detection algorithm

Input : A sequence of undirected graphs: $\{G_1, G_2, \ldots G_T\}$
Output: Community-based anomalies and the discovery timestamps

for every graph G_i in the sequence **do**
  /* Detect graph representatives */
  $Rep(G_i) = SV(G_{i-1})$;
  $SV(G_i) = \emptyset$;
  for every node $v_j \in G_i$ **do**
    if $v_j \in G_{i+1}$ **then**
      add v_j to $Rep(G_i)$;
      add v_j to $SV(G_i)$;
    end
  end
  /* Enumerate communities */
  CommunityEnumeration($Rep(G_i)$);
  Create community list CG_i;
end
/* Detect community representatives */
for every graph G_i in the sequence **do**
  for every community C_t^i **do**
    if $v_j \in C_t^i$ **then**
      Add i to the list $VC_t^{v_j}$;
      $NC_t^{v_j} = NC_t^{v_j} + 1$;
    end
  end
end
/* Establish community relationship */
for every community $C_t^i \in CG_t$ **do**
  Choose one node $v_j \in C_t^i$ with minimum $NC_t^{v_j}$ value ;
  Add v_j to $Checked(G_t)$;
  for every k, where $k \in VC_{t+1}^{v_j}$ **do**
    if $(V(C_t^i) \subseteq V(C_{t+1}^k))$ OR $(V(C_t^i) \supset V(C_{t+1}^k))$ **then**
      Establish the relationship $C_t^i \rightarrow C_{t-1}^k$;
    end
  end
  for every k, where $k \in VC_{t-1}^{v_j}$ **do**
    if $((C_{t-1}^k \rightarrow C_t^i) = FALSE)$ AND $(\left|C_{t-1}^k\right| > \left|C_t^i\right|)$ AND $(v_j \notin Checked(G_{t-1}))$ **then**
      if $V(C_t^i) \subset V(C_{t-1}^k)$ **then**
        Establish the relationship $C_{t-1}^k \rightarrow C_t^i$;
      end
    end
  end
end
/* Use decision rules to detect the anomalies */
for every community C_t^i in graph sequence **do**
  if C_t^i is an anomalous community based on decision rules **then**
    Output the community C_t^i with its anomaly type and discovery time t;
  end
end

Table 6 Performance comparison on synthetic data

Dataset	T_{non}(ms)	T_{rep}(ms)	Born	Vanished	Grown	Shrunken	Merged	Split
syn_500	265	25	3,425	3,482	87	79	18	23
syn_1000_1	1,132	87	8,702	8,569	154	119	42	33
syn_1500	6,442	329	23,482	23,621	401	295	71	86
syn_2000	16,182	489	23,111	23,178	333	278	71	83
syn_3000	61,912	1,354	59,261	59,220	813	718	465	313

on the datasets *syn_500*, *syn_1000_1*, *syn_1500*, *syn_2000*, and *syn_3000* are shown in Table 6, where T_{non} is the runtime of the non–representative algorithm, T_{rep} is the runtime of representative algorithm, and the last six columns are the counts of the six types of anomalies detected by the algorithm. From Fig. 12, we can see that the representative-based algorithm achieves a speedup of 11–46 times with respect to the non–representative-based algorithm. Additionally, the experimental results show that our algorithm is scalable to large graphs.

In the second experiment, we try to test the sole effectiveness of graph representatives in the community enumeration step. We use our in-house parallel MCE algorithm (Schmidt et al. 2009) (available upon request) to enumerate the communities in each graph for both algorithms. However, as discussed in Section 4.3, we enumerate all the communities in each graph for the non–representative-based method, but in the representative-based algorithm, we use the graph representatives as seeds to avoid graph-specific community enumeration.

The results are shown in Table 7, where NC_{non} is the number of cliques enumerated by the non–representative-based method, NC_{rep} is the number of cliques enumerated by the representative-based method, TC_{non} is the runtime of community enumeration using the non–representative-based method, and TC_{rep} is the runtime of community enumeration using representative-based method.

As shown in Table 7, the representative-based method achieves speedups of more than 1.1 in community enumeration on the dataset *syn_1000_1*, in which 10% of the vertices are graph-specific vertices; speedups of around 1.4 on the dataset *syn_1000_2*, in which 20% of the vertices are graph-specific vertices; and speedups of around 2 on the dataset *syn_1000_3*, in which 30% of the vertices are graph-specific vertices. The experiments on the datasets *syn_500*, *syn_1500*, *syn_2000*, and *syn_3000*

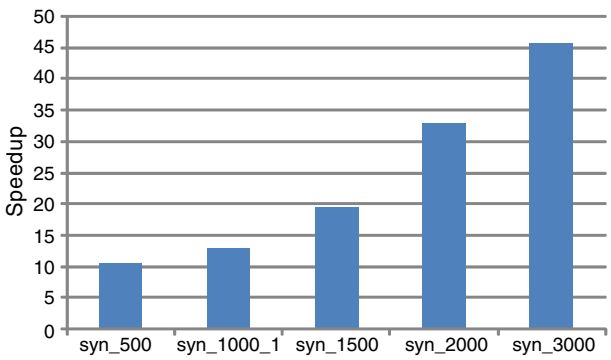

Fig. 12 Runtime speedup of the representative-based algorithm over the non–representative-based algorithm. The time to perform I/O operations is excluded

Table 7 Effectiveness of graph representatives

Dataset	E_i	NC_{non}	NC_{rep}	$TC_{\text{non}}(ms)$	$TC_{\text{rep}}(ms)$	Speedup
syn_1000_1	400,000	2,505	2,214	9	7.9	1.14
	550,000	2,541	2,299	9.4	8.4	1.12
	450,000	2,721	2,336	9.8	8.4	1.17
	600,000	2,564	2,303	9.3	8.3	1.12
	500,000	2,661	2,341	9.7	8.4	1.15
syn_1000_2	400,000	2,505	1,773	9	6.3	1.43
	550,000	2,541	1,878	9.4	6.9	1.36
	450,000	2,721	1,882	9.8	6.7	1.46
	600,000	2,564	1,880	9.3	6.7	1.39
	500,000	2,661	1,871	9.7	6.7	1.45
syn_1000_3	400,000	2,505	1,197	9	4.3	2.09
	550,000	2,541	1,382	9.4	5	1.83
	450,000	2,721	1,158	9.8	4.1	2.39
	600,000	2,564	1,221	9.3	4.4	2.11
	500,000	2,661	1,278	9.7	4.6	2.11

also show that the representative-based method can achieve a speedup of at least 1.1 in the community enumeration step, when the dataset has 10% graph-specific vertices.

6 Conclusion

In this paper, we have defined a new type of "in-disguise" anomaly, the *community-based anomaly*. In addition, we have proven that there are only six possible types of community-based anomalies in evolutionary networks: *grown*, *shrunken*, *merged*, *split*, *born*, and *vanished* communities. We have proposed a new method based on graph representatives and community representatives to reduce the computational cost. Based on the abnormal community decision rules, our algorithm can discover meaningful results in evolutionary networks that cannot be detected by other graph-based anomaly detection algorithms. The main properties of our algorithm are as follows:

- it is parameter-free and automatic by nature;
- it is applicable to evolutionary networks characterized by overlapping communities; and
- it is scalable to large networks.

We have demonstrated the effectiveness of our algorithm over a number of synthetic as well as practical examples. Experimental results on real-world networks show that our algorithm can detect meaningful community abnormalities.

Acknowledgements The authors would like to thank Matthew C. Schmidt for his maximal clique enumeration program code, and we would like to thank Kevin A. Wilson and Ye Jin for valuable discussions.

Appendix

Proofs for Theorems and Lemmas of Section 4.1

Lemma 1 *If community C_t^i has more than one predecessor (or successor), the sizes of its predecessors (or successors) are either all larger than $|C_t^i|$ or all smaller than $|C_t^i|$.*

Proof Suppose otherwise, that C_t^1 has a predecessor with smaller size, as well as one with a larger size. Let C_{t-1}^1, C_{t-1}^2,..., C_{t-1}^n (where $n \geq 2$) be all predecessors of C_t^i, and suppose that $|C_{t-1}^j| < |C_t^i|$ and $|C_{t-1}^k| > |C_t^i|$ for some $1 \leq j, k \leq n$, $j \neq k$. From Definition 5 and the sizes of the three communities, we know that $C_{t-1}^j \subset C_t^i$ and $C_t^i \subset C_{t-1}^k$, so $C_{t-1}^j \subset C_{t-1}^k$. However, C_{t-1}^j and C_{t-1}^k are both maximal cliques in the same graph, and $C_{t-1}^j \subset C_{t-1}^k$ contradicts the definition of a maximal clique. Therefore, it is impossible to have the size of one predecessor be larger than the size of the community and the size of another predecessor be smaller than the size of the community. □

Theorem 1 *Let G_t and G_{t+1} both be simple, undirected graphs, where communities are defined as maximal cliques. If G_{t+1} is the perturbed graph formed by either adding edges/nodes to or removing edges/nodes from the baseline graph G_t, then there are only six possible types of community-based anomalies between G_t and G_{t+1}: grown communities, shrunken communities, merged communities, split communities, born communities, and vanished communities, as defined in Definition* 7.

Proof Assume that $C_t^1, C_t^2, \ldots, C_t^m$ are all communities in G_t and that $V_t^1, V_t^2, \ldots, V_t^m$ are the node sets of the communities, respectively. Also assume that $C_{t+1}^1, C_{t+1}^2, \ldots, C_t^n$ are all communities in G_{t+1} and that $V_{t+1}^1, V_{t+1}^2, \ldots, V_{t+1}^n$ are the node sets of the communities, respectively. Here, we define $V_t^i = V_{t+1}^j$ to mean that V_t^i only contains all the nodes in V_{t+1}^j.

To determine the type of a specific community, we only need to compare the node sets of communities in G_{t+1} with the node sets of communities in G_t. If $V_{t+1}^j = V_t^i$, where $1 \leq i \leq m$ and $1 \leq j \leq n$, then community C_{t+1}^j contains exactly those nodes in community C_t^i, which means that C_{t+1}^j is a conserved community and not an anomaly.

In the following, we consider all possible anomalies by analyzing all possible mappings between predecessors and successors. In particular, when deciding if community C_t^i is an anomaly, we do not need to consider the situation where C_t^i has a single successor as long as we have covered all cases for the predecessors of C_{t+1}^j. If the community C_t^i has only one successor C_{t+1}^j, then the community C_{t+1}^j should have either one predecessor or more than one predecessor, both of which can be covered by using predecessor conditions. The same reasoning applies for not considering the case where a community has more than one successor of larger size. In other words, we need to consider all cases for predecessors, but only two cases for successors: when

a community has no successor and when a community has more than one successor of smaller size.

1. For a specific j (where $1 \leq j \leq n$), there is at least one i (where $1 \leq i \leq m$) that satisfies $V_{t+1}^{j} \subset V_{t}^{i}$. Then, by Definition 5, community C_{t+1}^{j} has at least one predecessor, including C_{t}^{i}, with larger size than C_{t+1}^{j}. Let $I = \{i \mid V_{t+1}^{j} \subset V_{t}^{i}\}$. There are two non-exclusive sub-cases here:
   (a) For $\ell \in I$, if there is some k (where $1 \leq k \leq n$) other than j that satisfies $V_{t+1}^{k} \subset V_{t}^{\ell}$, then C_{t}^{ℓ} has more than one smaller-size successor (C_{t+1}^{j} and C_{t+1}^{k}). Additionally, by Lemma 1, we know that C_{t}^{ℓ} cannot have a successor with larger size than C_{t}^{j}. Thus, C_{t}^{ℓ} is a split community, and C_{t+1}^{j} is one of its products.
   (b) For $\ell \in I$, if there is no k (where $1 \leq k \leq n$) other than j that satisfies $V_{t+1}^{k} \subset V_{t}^{\ell}$, then C_{t}^{ℓ} has only one smaller-size successor C_{t+1}^{j}, and C_{t+1}^{j} has at least one predecessor, including C_{t}^{ℓ}, with larger size. Also, by Lemma 1, we know that C_{t+1}^{j} cannot have a predecessor with smaller size than C_{t+1}^{j}. Thus, C_{t+1}^{j} is a shrunken community.
2. For a specific j (where $1 \leq j \leq n$), there is only one i (where $1 \leq i \leq m$) that satisfies $V_{t+1}^{j} \supset V_{t}^{i}$. Then, community C_{t+1}^{j} has one predecessor C_{t}^{i} with smaller size than C_{t+1}^{j}. Additionally, by Lemma 1, we know that C_{t+1}^{j} cannot have a predecessor with larger size than C_{t+1}^{j}. Thus, community C_{t+1}^{j} is a grown community.
3. For a specific j (where $1 \leq j \leq n$), there is more than one i (where $1 \leq i \leq m$) that satisfies $V_{t+1}^{j} \supset V_{t}^{i}$. Then, community C_{t+1}^{j} has more than one predecessor with smaller size. Also, by Lemma 1, we know that C_{t+1}^{j} cannot have a predecessor with larger size than C_{t+1}^{j}. Thus, community C_{t+1}^{j} is a merged community.
4. For a specific j (where $1 \leq j \leq n$), there is no i (where $1 \leq i \leq m$) that satisfies $V_{t+1}^{j} \supset V_{t}^{i}$ or $V_{t+1}^{j} \subset V_{t}^{i}$, which means that community C_{t+1}^{j} has no predecessor. Thus, C_{t+1}^{j} is a born community.
5. For a specific i (where $1 \leq i \leq m$), there is at least one j (where $1 \leq j \leq n$) that satisfies $V_{t+1}^{j} \subset V_{t}^{i}$. Let $J = \{j \mid V_{t+1}^{j} \subset V_{t}^{i}\}$. Then, for each $k \in J$, there is at least one i (where $1 \leq i \leq m$) that satisfies $V_{t+1}^{k} \subset V_{t}^{i}$, which is case 1. Thus, this case can be converted to case 1.
6. For a specific i (where $1 \leq i \leq m$), there is at least one j (where $1 \leq j \leq n$) that satisfies $V_{t+1}^{j} \supset V_{t}^{i}$. Let $J = \{j \mid V_{t+1}^{j} \supset V_{t}^{i}\}$. Then, for each $k \in J$, there is at least one i (where $1 \leq i \leq m$) that satisfies $V_{t+1}^{k} \supset V_{t}^{i}$, which is case 2 or 3. Thus, this case can be converted to case 2 or 3.
7. For a specific i (where $1 \leq i \leq m$), there is no j (where $1 \leq j \leq n$) that satisfies $V_{t+1}^{j} \supset V_{t}^{i}$ or $V_{t+1}^{j} \subset V_{t}^{i}$, which means that community C_{t}^{i} has no successor. Thus, C_{t}^{i} is a vanished community.

Since all relationships between V_{t+1}^{j} (where $1 \leq j \leq n$) and V_{t}^{i} (where $1 \leq i \leq m$) have been covered, there are only six possible different types of community-based anomalies. □

Theorem 2 *If community C_t^i is represented by vertex $v_i \in C_t^i$, and community C_{t+1}^j is represented by vertex $v_j \in C_{t+1}^j$, where $C_t^i \to C_{t+1}^j$, then $v_i \in C_{t+1}^j$ or $v_j \in C_t^i$.*

Proof By Definition 5, $C_t^i \to C_{t+1}^j$ implies that $C_t^i \subseteq C_{t+1}^j$ or $C_{t+1}^j \subseteq C_t^i$. If $C_t^i \subseteq C_{t+1}^j$, then $v_i \in C_{t+1}^j$, and if $C_{t+1}^j \subseteq C_t^i$, then $v_j \in C_t^i$. □

References

Bader, D. A., & Madduri, K. (2006). *Gtgraph: A synthetic graph generator suite*. Technical Report GA 30332, Georgia Institute of Technology, Atlanta.

Baird, D., & Ulanowicz, R. E. (1989). The seasonal dynamics of the chesapeake bay ecosystem. *Ecological Monographs, 59*, 329–364.

Chakrabarti, D. (2004). Autopart: Parameter-free graph partitioning and outlier detection. In *PKDD* (pp. 112–124).

Chakrabarti, D., Zhan, Y., & Faloutsos, C. (2004). R-mat: A recursive model for graph mining. In *SDM*.

Chan, P. K., & Mahoney, M. V. (2005). Modeling multiple time series for anomaly detection. In *ICDM* (pp. 90–97).

Chen, L., DeVries, A. L., & Cheng, C. H. (1997). Convergent evolution of antifreeze glycoproteins in Antarctic notothenioid fish and Arctic cod. *Proceedings of the National Academy of Sciences of the United States of America, 94*, 3817–3822.

Cheng, H., Tan, P.-N., Potter, C., & Klooster, S. (2008). A robust graph–based algorithm for detection and characterization of anomalies in noisy multivariate time series. In *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2008* (pp. 349–358).

Clauset, G., Newman, M. E., & Moore, C. (2004). Finding community structure in very large networks. *Physical Review E, 70*, 1–6.

Eberle, W., & Holder, L. (2006). Detecting anomalies in cargo shipments using graph properties. In *Proceedings of the IEEE intelligence and security informatics conference*.

Eberle, W., & Holder, L. (2007). Discovering structural anomalies in graph–based data. In *Workshops proceedings of the 7th IEEE international conference on data mining* (pp. 393–398).

Girvan, M., & Newman, M. E. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences, 99*(12), 7821–7826.

Hautamäki, V., Kärkkäinen, I., & Fränti, P. (2004). Outlier detection using k-nearest neighbour graph. In *ICPR (3)* (pp. 430–433).

Hopcroft, J., Khan, O., Kulis, B., & Selman, B. (2004). Tracking evolving communities in large linked networks. *Proceedings of the National Academy of Sciences, 101*, 5249–5253.

Keogh, E. J., Lin, J., & Fu, A. W.-C. (2005). Hot sax: Efficiently finding the most unusual time series subsequence. In *ICDM* (pp. 226–233).

Lin, S., & Chalupsky, H. (2003). Unsupervised link discovery in multi-relational data via rarity analysis. In *ICDM* (pp. 171–178).

Long, M., Betran, E., Thornton, K., & Wang, W. (2003). The origin of new genes: Glimpses from the young and old. *Nature Reviews. Genetics, 4*(11), 865–875.

Moonesinghe, H., & Tan, P.-N. (2006). Outlier detection using random walks. In *International Conference on Tools with Artificial Intelligence, ICTAI* (pp. 532–539).

Noble, C. C., & Cook, D. J. (2003). Graph–based anomaly detection. In *KDD '03: Proceedings of the 9th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 631–636). New York: ACM.

Padmanabh, K., Vanteddu, A., Sen, S., & Gupta, P. (2007). Random walk on random graph based outlier detection in wireless sensor networks. In *Wireless communication and sensor networks* (pp. 45–49).

Palla, G., Derenyi, I., Farkas, I., & Vicsek, T. (2005). Uncovering the overlapping community structure of complex networks in nature and society. *Nature, 435*(7043), 814–818.

Palla, G., Albert-László Barabási, A., & Vicsek, T. (2007). Quantifying social group evolution. *Nature, 446*, 664–667.

Schmidt, M. C., Samatova, N. F., Thomas, K., & Park, B.-H. (2009). A scalable, parallel algorithm for maximal clique enumeration. *Journal of Parallel and Distributed Computing, 69*(4), 417–428.

Shetty, J., & Adibi, J. (2005). Discovering important nodes through graph entropy the case of enron email database. In *LinkKDD '05: proceedings of the 3rd international workshop on link discovery* (pp. 74–81). New York: ACM.

Snel, B., Bork, P., & Huynen, M. A. (2000). Genome evolution. Gene fusion versus gene fission. *Trends in Genetics, 16*, 9–11.

Staniford-chen, S., Cheung, S., Crawford, R., Dilger, M., Frank, J., Hoagl, J., et al. (1996). Grids—a graph based intrusion detection system for large networks. In *Proceedings of the 19th national information systems security conference* (pp. 361–370).

Steinhaeuser, K., Chawla, N. V., & Ganguly, A. R. (2009). An exploration of climate data using complex networks. In *SensorKDD '09: Proceedings of the 3rd international workshop on knowledge discovery from sensor data* (pp. 23–31). New York: ACM.

Sun, J., Faloutsos, C., Papadimitriou, S., & Yu, P. S. (2007). Graphscope: Parameter-free mining of large time-evolving graphs. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 687–696). San Jose: ACM.

Sun, J., Qu, H., Chakrabarti, D., & Faloutsos, C. (2005). Neighborhood formation and anomaly detection in bipartite graphs. In *The 5th IEEE International Conference on Data Mining (ICDM)* (pp. 418–425).

Sun, J., Tao, D., & Faloutsos, C. (2006). Beyond streams and graphs: dynamic tensor analysis. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 374–383). New York: ACM.

Tantipathananandh, C., Wolf, T. B., & Kempe, D. (2007). A framework for community identification in dynamic social networks. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 717–726). ACM.

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature, 393*(6684), 440–442.

Zhang J. (2003). Evolution by gene duplication: An update. *Trends in Ecology & Evolution, 18*, 292–298.