Detecting and Tracking Community Dynamics in Evolutionary Networks

Zhengzhang Chen*†, Kevin A. Wilson*‡, Ye Jin*†, William Hendrix*† and Nagiza F. Samatova*†§
*Department of Computer Science
North Carolina State University, Raleigh, NC, USA
†Computer Science and Mathematics Division
Oak Ridge National Laboratory, Oak Ridge, TN, USA
‡Research Computing Division, RTI International
Research Triangle Park, NC, USA
§Corresponding author: samatovan@ornl.gov

***Abstract*—Community structure or clustering is ubiquitous in many evolutionary networks including social networks, biological networks and financial market networks. Detecting and tracking community deviations in evolutionary networks can uncover important and interesting behaviors that are latent if we ignore the dynamic information. In biological networks, for example, a small variation in a gene community may indicate an event, such as gene fusion, gene fission, or gene decay. In contrast to the previous work on detecting communities in static graphs or tracking conserved communities in time-varying graphs, this paper first introduces the concept of community dynamics, and then shows that the baseline approach by enumerating all communities in each graph and comparing all pairs of communities between consecutive graphs is infeasible and impractical. We propose an efficient method for detecting and tracking community dynamics in evolutionary networks by introducing graph representatives and community representatives to avoid generating redundant communities and limit the search space. We measure the performance of the representative-based algorithm by comparison to the baseline algorithm on synthetic networks, and our experiments show that our algorithm achieves a runtime speedup of 11–46. The method has also been applied to two real-world evolutionary networks including Food Web and Enron Email. Significant and informative community dynamics have been detected in both cases.**

***Keywords*-community detection; evolutionary analysis; social networks; community dynamics;**

I. Introduction

Networks of dynamic systems can be highly clustered, or form community structures [18]. A community, defined as a collection of individual objects that interact unusually frequently, is a very common substructure in many networks [7], 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 [7]. 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.

Many algorithms have been developed for detecting community structures in static graphs. Girvan and Newman [7] 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.* [6] proposed a modularity-based algorithm. In contrast, Palla and Derenyi [12] did not focus on detecting separate communities, but on finding overlapping communities. Defining communities as maximal cliques, Schmidt *et al.* [14] 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 and Khan [9] proposed a method which utilizes a "nature community" to track stable clusters over time. A framework for identifying communities in dynamic social networks, proposed by Chayang and Berger-Wolf [17], makes explicit use of temporal changes. Using the Clique Percolation Method to locate communities, Palla *et al.* [13] defined auto-correlation and stationarity to characterize a community. They also introduced six basic community events, which will be used in our paper to define the community dynamics. But instead of focusing on the stationarity of the communities, our goal is to detect and track community dynamics. From an application perspective, Steinhaeuser *et al.* [16] provided a method to identify climate regions by detecting communities in time-varying climate networks.

However, communities in real networks change over time. In contrast to the previous work on tracking *conserved* communities, the evolution of structure within social networks has been addressed in [10]. Kumar *et al.* presented a model of network growth to capture singletons, isolated communities, and a giant component. Backstrom *et al.* [1] also considered the structural changes of social networks, but instead of detecting isolated communities, they focused on the structural features that influence the group growth for a period of time. In this paper, we not only consider grown communities, but also detect all the other types of community dynamics, including shrunken communities, merged communities, split communities, born communities,

and vanished communities (see Definition 6).

Being able to detect all types of small community deviations can help us understand and exploit these networks more effectively. For example, in biological networks, a small variation in a gene community may represent an event, such as gene fusion [15], gene fission [15], gene gain [5], gene decay [11], or gene duplication [19], 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 can be seen in Section IV-B.

Our approach follows from the need to address the following four issues:

- How do we define community dynamics, and how many types of community dynamics are possible in evolutionary networks? Community dynamics would reveal latent behaviors of the network, as opposed to conserved communities or communities in a single snapshot. For example, is there any community in snapshot t that splits into smaller communities or merges with others in snapshot k? Does any community in snapshot t disappear in snapshot k, or does any new community appear in snapshot k? Do the sizes of the communities change over time?
- Most real networks are dynamic and characterized by overlapping communities [12]. Detecting community dynamics from networks characterized by overlapping communities is more challenging than discovering separated communities in static networks.
- How do we detect and track community dynamics across multiple graphs? As we mentioned earlier, real-world networks change over time, requiring us to adopt evolutionary analysis techniques to detect and track these community dynamics.
- Since there may be hundreds or even thousands of communities in each real-world network, our algorithm needs to be able to scale to large graphs.

A naïve solution to these problems is to first enumerate all communities in each graph, and then compare all pairs of communities between two consecutive graphs. This two-stage approach is infeasible and impractical because of the possible exponential number of communities and huge search space. For example, if we consider a maximal clique as a community, a graph can have up to $3^{\frac{n}{3}}$ communities, where n is the number of vertices, and the possible pairs of communities between two consecutive graphs could be in the order of $3^{\frac{2n}{3}}$. Meanwhile, there are many redundant communities (e.g., graph-dependent communities (see Definition 3)) in each graph, and it does not make much sense to compare pairs of communities that contain no common members or few members.

Therefore, in this paper, we propose an efficient algorithm based on the proposed notion of *graph representatives* and *community representatives*. Graph representatives can help us reduce the expensive computational cost caused by enumerating communities, which are modeled as maximal cliques in our paper. Community representatives can be exploited to identify the community dynamics.

The contributions of the paper are:

1) We prove that only six types of dynamic communities are possible in dynamic simple undirected graphs;
2) We develop an efficient algorithm for detecting and tracking community dynamics based on the introduced concept of *graph representatives* and *community representatives*; and
3) We evaluate our method on real datasets to confirm its applicability in practice.

II. Problem Statement

Table I
Symbol Table

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

In this paper, the ultimate goal is to detect and track community dynamics in evolutionary networks, and our algorithm is based on graph representatives and community representatives. Thus, the following terms and problems need to be addressed. The symbols used throughout the paper are listed in Table I.

Problem 1 (Detecting and Tracking Community dynamics): Given a time-varying graph sequence $\mathcal{G} = \{G_1, G_2, G_3, ...\}$, detect and track the community dynamics, including grown communities, shrunken communities, merged communities, split communities, born communities, and vanished communities (see Definition 6).

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

There is no formal definition for the community structure in a network [7]. The simplest and the most conservative definition of a community is a clique, a set of vertices that

are adjacent to one another. Another definition used by Newman [7] is a dense subgraph, or a group of vertices within which the connections are denser than between different groups. Because our goal is to detect community dynamics, we prefer the more specific definition (clique) rather than the fuzzy one (dense subgraph). More importantly, from an application perspective, we may lose important changing trend information by using the dense subgraph definition. For example, consider protein functional modules (biological communities) in protein-protein interaction networks. Two overlapping communities in one organism may merge in another organism to affect a particular phenotype (e.g. hydrogen/ethanol production, high temperature resistance, nitrogen fixation, or aerobic respiration). If we consider a dense subgraph as a community, then the merging event for two overlapping communities will become hard or even impossible to detect. Also, our target applications necessitate a simpler and more stringent definition of a community—a clique.

This form of abstraction has been found beneficial in various application domains, such as those in biology. Specifically, such an abstraction was found to be less sensitive to noise inherent to biological networks and better at uncovering biologically relevant protein complexes [8]. Moreover, in the context of community dynamics, a small perturbation to the community structure, such as a single edge or vertex deletion, may indicate important events, such as gene fusion [15], gene fission [15], gene gain [5], gene decay [11], or gene duplication [19]. While useful in some application domains, clique-based modeling of a community might be inadequate in other application domains such as those in social networks.

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

Definition 3 (Graph representative): Representatives of graph G_t are the nodes that also appear in G_{t-1}, G_{t+1}, or both, thus $Rep(G_t) = \{v_i \mid v_i \in G_t \bigcap G_j, where\, |j-t| = 1\}$.

Nodes that only appear in one graph are called graph-dependent nodes/vertices. If a community only contains graph-dependent vertices, then it can be considered as a "graph-dependent" community, which isn't a community dynamic we try to detect. Thus, using graph representatives as seeds, we do not need to enumerate all communities in the graphs, but only the communities containing the representatives. This technique can potentially reduce computational time (see Section III for details).

Definition 4 (Community predecessor and successor): If one community C_t^i at snapshot t is the subset or superset of the 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. And the community relationship is defined by $C_t^i \rightarrow C_{t+1}^j$.

Definition 5 (Community representative): 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 III for details).

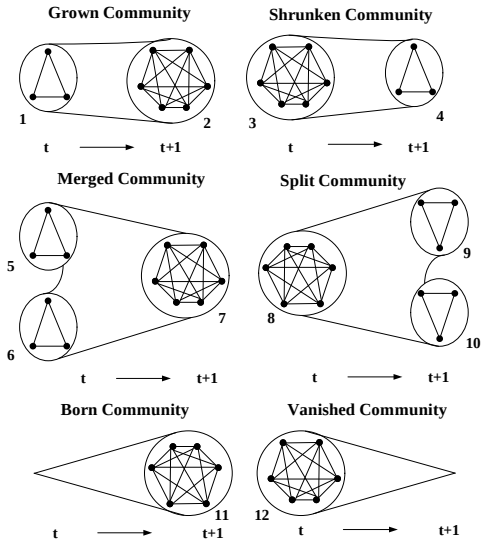

Figure 1. Possible types of community dynamics in evolutionary networks.

Definition 6 (Community dynamics): In contrast to [13], which focuses on the stability/stationarity of the communities, our goal is to detect and track community dynamics. As there are six basic events that may occur to a community [13], we can define six possible types of community dynamics in evolutionary networks (see Fig. 1).

1) *Grown community*
   In real-world networks, some "large" communities, like community 2, are grown from previous "smaller" communities by adding some new members.
2) *Shrunken community*
   On the other hand, the shrunken communities, like community 4, are the communities shrunk from previous "bigger" communities by losing some members.
3) *Merged community*
   In addition, often two or more "small" communities at snapshot t join together to be one merged community, like community 7, at snapshot $t+1$.
4) *Split community*

Meanwhile, a split community, like community 8, at snapshot t are broken up into more than one different communities at snapshot $t+1$.

5) *Born community*
What's more, some "new" communities, like community 11, would appear in some snapshots.
6) *Vanished community*
Finally, some "old" communities, like community 12, would disappear.

III. Representative-based Algorithm for Detecting and Tracking Community Dynamics

In this section, we first describe some necessary lemmas and theories. Then, based on the dynamic community decision rules described in Section III-B, we illustrate how to detect dynamic communities in SectionIII-C.

A. Lemmas and Theorems

Lemma 1: If community C_t^i has more than one predecessor, the sizes of its predecessors are either all larger than $\left|C_t^i\right|$ or all smaller than $\left|C_t^i\right|$.

Proof: Suppose otherwise, that, 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 there is one predecessor C_{t-1}^j with $\left|C_{t-1}^j\right| \leq \left|C_t^i\right|$ and another predecessor C_{t-1}^k $\left|C_{t-1}^k\right| \geq \left|C_t^i\right|$ for some $1 \leq j,k \leq n$, $j \neq k$. From Definition 4 and the sizes of the three communities, we know that $C_{t-1}^j \subseteq C_t^i$ and $C_t^i \subseteq C_{t-1}^k$, so $C_{t-1}^j \subseteq C_{t-1}^k$. However, C_{t-1}^j and C_{t-1}^k, where $j \neq k$, are two different cliques in the same graph, so $C_{t-1}^j \subset C_{t-1}^k$. In addition, C_{t-1}^j and C_{t-1}^k are both maximal cliques in the same graph, so $C_{t-1}^j \subset C_{t-1}^k$ contradicts the definition of a maximal clique. Therefore, it is impossible to have one predecessor with larger size than the community, and another predecessor with smaller size. ■

Similarly, we can prove that if a community C_t^i has more than one successor, the sizes of its successors are either all larger than $\left|C_t^i\right|$ or all smaller than $\left|C_t^i\right|$.

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 dynamics between G_t and G_{t+1}: grown communities, shrunken communities, merged communities, split communities, born communities, and vanished communities, as defined in Definition 6.

Based on Lemma 1, it is straightforward to prove Theorem 1. But for the sake of completeness, we present our proof here.

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. 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 a community dynamic. However, for communities C_t^i and C_{t+1}^j that do not have a "match" in G_{t+1} or G_t, respectively, then there are overall seven possibilities.

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 4, 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). By Lemma 1, 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, 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, C_{t+1}^j has more than one predecessor with smaller size. Also, by Lemma 1, C_{t+1}^j cannot have a predecessor with larger size than C_{t+1}^j. Thus, 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 dynamics. ■

B. Decision Rules for Community Dynamics

The following decision rules for community dynamics hold based on Definition 6 and Theorem 1:

1) If community C_t^i has only one predecessor C_{t-1}^{j}:
   a) If the size of the predecessor is smaller than $\left|C_t^i\right|$, then C_t^i is a grown community.
   b) If the size of the predecessor is larger than $\left|C_t^i\right|$ 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 $\left|C_t^i\right|$ 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 $\left|C_t^i\right|$, then C_t^i is a merged community.
   b) If the sizes of the predecessors are all larger than $\left|C_t^i\right|$ 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 $\left|C_t^i\right|$ 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.

C. Algorithm Description

Non-representative-based Method:

To detect and track the community dynamics without using graph or community representatives, we first enumerate all communities in each graph, and then we compare all pairs of communities belonging to consecutive timestamps. For example, to find the successors of community A in Fig. 2, 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. For real-world networks with hundreds or thousands of communities, the non-representative-based method is very computationally expensive.

Representative-based Method:

Intuitively, we first find graph representatives in each graph (see Algorithm 1), and we enumerate the communities in each graph using the graph representatives as the seeds to avoid generating redundant communities. We then use community representatives to establish the relationship between the communities from different time stamps (see lines 10–19 in Algorithm 2). If all the predecessors and successors of the community C_t^i can be found, then we can determine whether C_t^i is a community dynamic by applying the decision rules (see Section III-B).

For example, in Fig. 3, we use the filled triangle or rectangle nodes as seeds to generate communities. We then look for community representatives (triangular nodes). Taking advantage of the community representatives, we track forward in the sequence to establish the community relationship between the two time stamps. How can we determine if one community at timestamp $t+1$ (or t) is the successor (or predecessor) of another community at t (or $t+1$)? Let us take community A at timestamp t, for example. To find the successor of community A, first we need to find all communities that contain the community representative of A at timestamp $t+1$. In this case, only community F contains the community representative. Then, we check whether community F meets the condition of successor of A (see Definition 4). Only if this is true, do we establish the relationship between A and F. When there are only grown, merged, born or vanished communities like community *A* grows to community *F*, communities *B* and *C* merge into *E*, community *D* emerges and community *F* disappears, we do not need to track back in the sequence. In cases of shrunken communities or split communities, we may need to track back. 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 $3 \notin G$. In this case, we need to use representatives 10 and 6 to track back to establish the relationships between D and I, and E and G. The workflow of the algorithm is shown in Fig. 4.

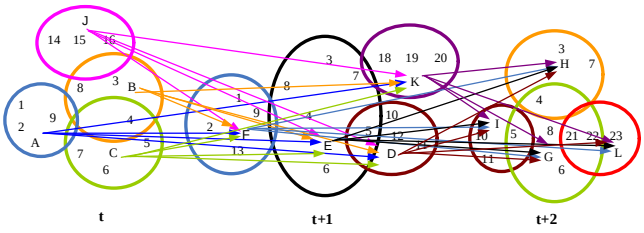

Figure 2. Example for tracking community dynamics using the non-representative-based method. (Best viewed in color.)

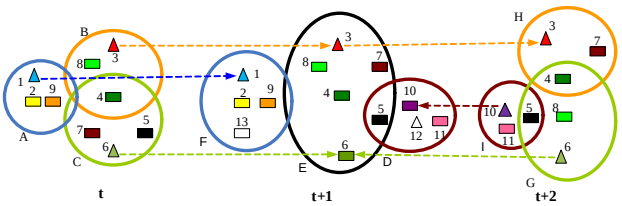

Figure 3. Example for tracking community dynamics using the representative-based method. Note that triangles are community representatives; filled shapes are graph representatives; empty shapes are graph-dependent vertices; circles are communities; and dashed lines are community relationships. (Best viewed in color.)

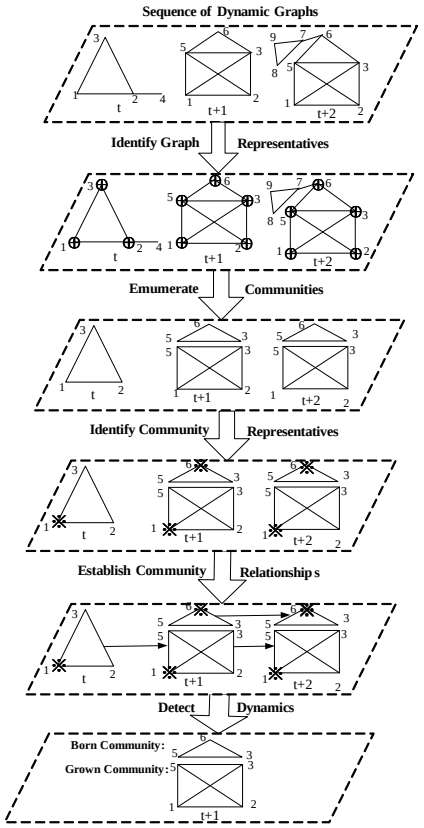

Figure 4. Workflow of the representative-based algorithm with three timestamps.

Algorithm 1: Graph Representative Detection

Input : Graphs G_t and G_{t+1}
Output: $Rep(G_t)$ - Community representatives of G_t;
$SV(G_t)$ - Common nodes in both G_i and G_{t+1}

$Rep(G_t) = SV(G_{t-1})$;
$SV(G_t) = \emptyset$;
for *every node* $v_j \in G_t$ **do**
  if $v_j \in G_{t+1}$ **then**
    add v_j to $Rep(G_t)$;
    add v_j to $SV(G_t)$;

Algorithm 2: Detecting and tracking community dynamics algorithm

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

/* Communities Enumeration */
for *every graph* G_t *in the sequence* **do**
  $Rep(G_t)$ = RepresentativeDetection(G_t);
  CommunityEnumeration($Rep(G_t)$);
  Create community list CG_t;
/* Detect community representatives */
for *every graph* G_t *in the sequence* **do**
  for *every community* C_t^i **do**
    for *every node* $v_j \in C_t^i$ **do**
      Add i to the list $VC_t^{v_j}$;
      $NC_t^{v_j} = NC_t^{v_j} + 1$;
/* 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 $C_t^i \rightarrow C_{t-1}^k$;
  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 $C_{t-1}^k \rightarrow C_t^i$;
/* Community dynamics detection */
for *every community* C_t^i *in graph sequence* **do**
  Use decision rules to decide the type of community dynamics;

IV. Experiments

In this section, we evaluate our algorithm on both synthetic and real-world datasets. We focus on answering the following two questions:

1) What is the performance of our algorithm using the representative-based technique?
2) Is our algorithm scalable to large graphs?
3) Does the algorithm provide insights into real-world datasets?

Section IV-A studies the performance of the representative-based algorithm relative to the baseline algorithm on synthetic networks. Section IV-B describes our experiments on real networks, including the Food Web and Enron Email datasets. 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. 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 into the runtime of our algorithm.

A. Synthetic Experiments

In this experiment, we study the effectiveness of the proposed representative-based technique. All the graphs in the synthetic datasets are generated by GTgraph [2] and follow the Recursive Matrix Graph model (R-MAT) [4] so that they have a small-world nature. The parameters for the synthetic graphs, which appear in Table II, are defined as follows: $|V|$ is the number of vertices in a graph, Num_{gv} is the number of graph-dependent vertices in a graph, and E_i is the number of edges in a graph G_i. On all graphs, we use the default values of 0.45, 0.15, 0.15 and 0.25 for the R-MAT parameters a, b, c, d, respectively, with a : b and a : c ratios of 3:1, as in many real world graphs [4]. After graph enumeration, we use a program to relabel some of the vertices according to the parameter Num_{gv}, so that we can have some graph-dependent vertices in each graph when we build the graph sequence. For example, in the dataset *syn_500*, we can relabel the vertices $v_j \in [451, 500]$ in graph G_i as $v_j + 50 \times (i-1)$. Other graph-dependent vertices in other datasets can be similarly relabeled.

Table II
TABLE OF SYNTHETIC DATASETS

Dataset	$\|V\|$	Num_{gv}	E_1	E_2	E_3	E_4	E_5
syn_500	500	50	8000	11000	9000	12000	10000
syn_1000_1	1000	100	400000	550000	450000	600000	500000
syn_1000_2	1000	200					
syn_1000_3	1000	300					
syn_1500	1500	150	640000	880000	720000	960000	800000
syn_2000	2000	200	800000	1100000	900000	1200000	1000000
syn_3000	3000	300	1600000	2200000	1800000	2400000	2000000

After building the sequence of graphs, we use the parallel MCE algorithm [14] to enumerate the communities in the individual graphs, and apply the proposed algorithms to detect dynamic changes in the community (or clique) structure between the subsequent graphs. Namely, for the non-representative-based method, we enumerate *all* the communities in each graph, but for the representative-based algorithm, we use *graph representatives as seeds* to avoid graph-dependent community enumeration (see a more detailed discussion in Section III-C).

In the first experiment, we try to test the effectiveness of the graph representatives in the community enumeration step. The results appear in Table III: 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 III, the representative-based method achieves speedups of more than 1.1 on the dataset *syn_1000_1*, in which 10 percent of the vertices are graph-dependent vertices; speedups of around 1.4 on the dataset *syn_1000_2*, in which 20 percent of the vertices are graph-dependent vertices; and speedups of around 2 on the dataset *syn_1000_3*, in which 30 percent of the vertices are graph-dependent vertices. The experiments on the datasets *syn_500*, *syn_1500*, *syn_2000* and *syn_3000* 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 percent graph-dependent vertices.

Table III
TABLE OF EFFECTIVENESS OF GRAPH REPRESENTATIVES

Dataset	E_i	NC_{non}	NC_{rep}	$TC_{non}(ms)$	$TC_{rep}(ms)$	Speedup
syn_1000_1	400000	2505	2214	9	7.9	1.14
	550000	2541	2299	9.4	8.4	1.12
	450000	2721	2336	9.8	8.4	1.17
	600000	2564	2303	9.3	8.3	1.12
	500000	2661	2341	9.7	8.4	1.15
syn_1000_2	400000	2505	1773	9	6.3	1.43
	550000	2541	1878	9.4	6.9	1.36
	450000	2721	1882	9.8	6.7	1.46
	600000	2564	1880	9.3	6.7	1.39
	500000	2661	1871	9.7	6.7	1.45
syn_1000_3	400000	2505	1197	9	4.3	2.09
	550000	2541	1382	9.4	5	1.83
	450000	2721	1158	9.8	4.1	2.39
	600000	2564	1221	9.3	4.4	2.11
	500000	2661	1278	9.7	4.6	2.11

Table IV
TABLE OF SYNTHETIC EXPERIMENT RESULTS

Dataset	T_{non}(ms)	T_{rep}(ms)	Born	Vanished	Grown	Shrunken	Merged	Split
syn_500	265	25	3425	3482	87	79	18	23
syn_1000_1	1132	87	8702	8569	154	119	42	33
syn_1500	6442	329	23482	23621	401	295	71	86
syn_2000	16182	489	23111	23178	333	278	71	83
syn_3000	61912	1354	59261	59220	813	718	465	313

In the second experiment, we take community representatives into consideration and measure the entire runtime of the two algorithms for detecting the community dynamics across the five timestamps in each of the datasets. After community enumeration, the non-representative-based method would compare all pairs of communities belonging to two consecutive timestamps to build the community relationships, while the representative-based algorithm exploits community representatives to reduce the expensive computational cost of tracking communities. The result of the experiments on the datasets *syn_500*, *syn_1000_1*, *syn_1500*, *syn_2000* and

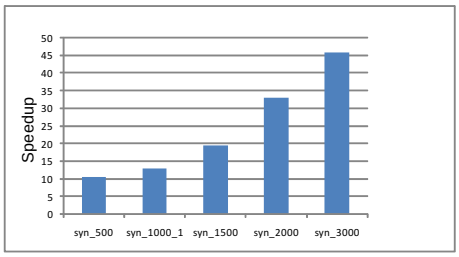

Figure 5. Runtime speedup of the representative-based algorithm over the non-representative-based algorithm for three synthetic datasets.

syn_3000 are shown in Table IV: 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 community dynamics detected by the algorithm. From Fig. 5, we can see that the representative-based algorithm achieves a speedup of 11–46 times with respect to the non-representative-based algorithm.

A time complexity analysis with regard to the number and types of community dynamics would be desirable, but it is hardly possible, because even a single node or edge change in a community may lead to tens of community changes at the same time.

B. Real-world Experiments

In addition to the synthetic experiments, we applied our algorithm to two real-world dynamic networks including Food Web and Enron Email. The minimum size of communities we considered here is three.

Food Web dataset: The food web consists of marine organisms living in the Chesapeake Bay, containing 33 vertices representing the ecosystem's most prominent taxa, which was originally compiled by Baird and Ulanowicz [3]. Edges between taxa denote trophic relationships—one taxon feeding on another. Here we ignore directionality and consider the network as an undirected graph. Newman [7] 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 dynamics in dynamic networks.

Table V
TABLE OF FOOD WEB COMMUNITIES

Season	Number of Communities	Community Dynamics
Spring	15	None
Summer	15	Four grown communities; One born community; Four communities will split in fall; One vanished community; one merged community
Fall	19	Two shrunken communities; four vanished communities (will disappear in winter)
Winter	9	Four merged communities

By applying our algorithm to the Food Web networks, we find all six types of community dynamics (see Table V). Summer is the most active community changing season: four communities grow because microzooplankton, which can not be found in spring, becomes involved in the energy flow network. Four communities split because bacteria do not feed on microzooplankton in fall. The disappearance of sea nettle in fall results in a vanished community (zooplankton, ctenophore, and sea nettle), which was a born community in summer. This indicates that the community is unstable. Due to lack of food in winter, four merged communities emerge in order to get the benefit from more members with the food energy. Typical community dynamic examples discovered in Food Web are shown in Fig. 6–7. We can see that Food Web is characterized by overlapping communities. Note that different cycles represent different communities at the same timestamp.

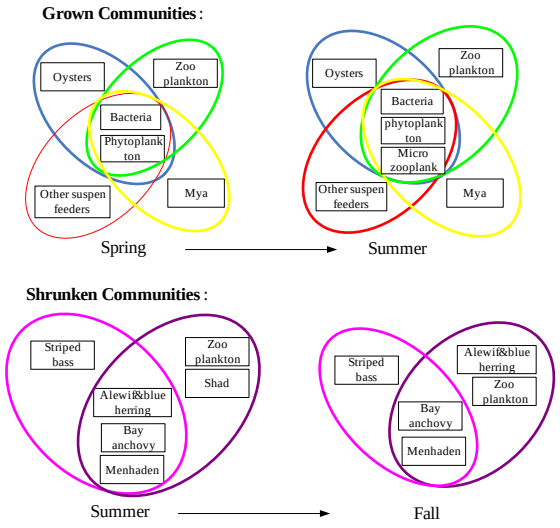

Figure 6. Grown community and shrunken community example of Food Web.

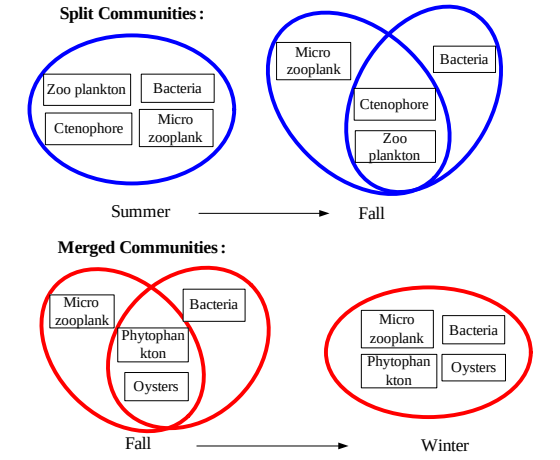

Figure 7. Split community and merged community example of Food Web.

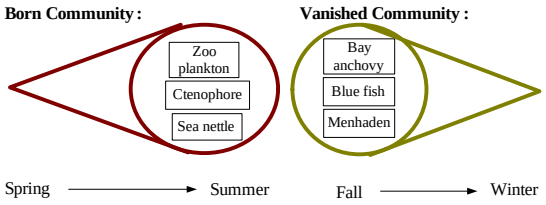

Figure 8. Born community and vanished community example of Food Web.

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 VI.

Table VI
ENRON EMAIL DATASET PROPERTIES

Month	Number of Edges	Number of Communities
Jan.	126	21
Feb.	190	56
Mar.	199	54
Apr.	240	66
May	273	90
Jun.	218	49
Jul.	240	68
Aug.	371	120
Sep.	343	110
Oct.	531	196
Nov.	438	143
Dec.	290	93

The community dynamics in each month discovered by our algorithm are shown in Table VII. We can see that there are more community dynamics 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 dynamics in October, let us consider one of the most important nodes—Louise Kitchen, the former President of Enron. There are 20 community dynamics containing Louise Kitchen in October: 16 born communities, 4 grown communities, 1 split community, and 1 shrunken community. From Fig. 9, 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 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.

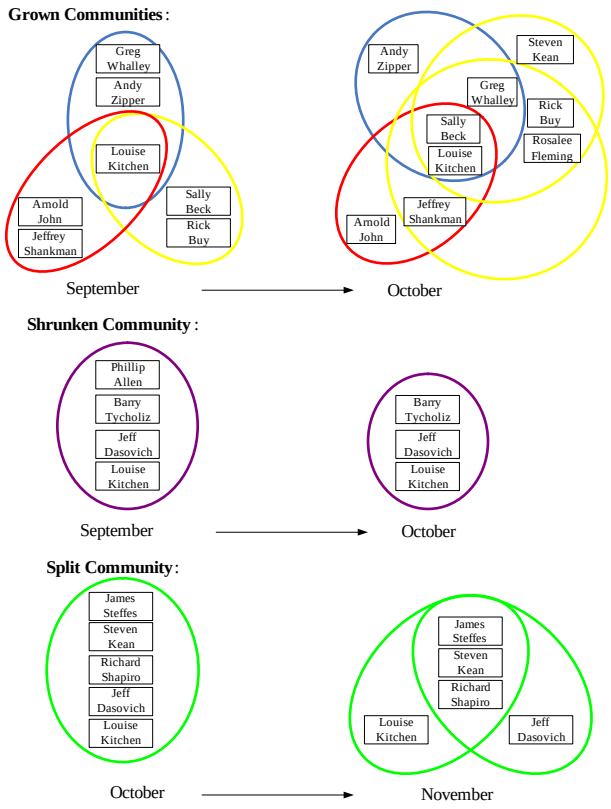

Figure 9. Community dynamics containing Louise Kitchen in October.

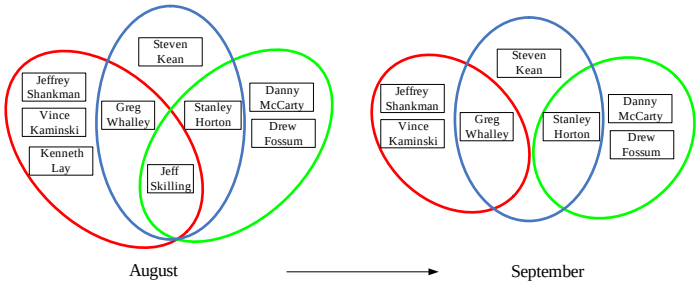

Figure 10. Shrunken communities due to Jeff Skillng resigning as CEO in August.

As a second example, take Jeff Skilling, the former CEO of Enron. There are 19 email communities containing Jeff Skilling in August, but among these, 3 communities shrank in September (see Fig. 10), 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.

V. CONCLUSION AND FUTURE WORK

In this paper, we proposed a new method for detecting and tracking all six types of community dynamics, including *grown*, *shrunken*, *merged*, *split*, *born*, and *vanished* communities, in evolutionary networks. Graph representatives

Table VII
COMMUNITY DYNAMICS IN ENRON EMAIL DATASET

Month	Grown	Shrunken	Merged	Split	Born	Vanished
Jan.	0	0	0	0	0	14
Feb.	3	1	0	1	48	32
Mar.	3	2	4	1	33	39
Apr.	6	1	0	2	42	43
May	3	4	0	1	75	76
Jun.	3	3	0	1	34	38
Jul.	1	0	2	0	58	54
Aug.	9	4	2	2	97	89
Sep.	8	9	3	1	79	56
Oct.	30	5	10	8	136	160
Nov.	7	13	0	9	102	97
Dec.	2	17	2	0	54	14

and community representatives are defined to reduce the computational cost of the algorithm. The main properties of the method are:

- Parameter-free and automatic;
- Effective and efficient; and
- Applicable to evolutionary networks even characterized with overlapping communities.

We have demonstrated the effectiveness of our method on a number of synthetic examples as well as real-world networks. Experimental results on real-world networks show that our algorithm can detect meaningful community dynamics, and do so in an efficient manner.

Because of lack of benchmark datasets, it is hardly possible to use detection accuracy as one of our metrics. In addition, a time window-based algorithm would be considered in our future work.

ACKNOWLEDGMENT

This work was funded by the Scientific Data Management Center under the Department of Energy's SciDAC program. The work of ZC was partially supported by the National Science Foundation grant NSF10-564. Oak Ridge National Laboratory is managed by UT-Battelle for the LLC U.S. D.O.E. under contract no. DEAC05-00OR22725.

REFERENCES

[1] BACKSTROM, L., HUTTENLOCHER, D., KLEINBERG, J., AND LAN, X. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (2006), ACM, pp. 44–54.

[2] BADER, D. A., AND MADDURI, K. Gtgraph: A synthetic graph generator suite, 2006.

[3] BAIRD, D., AND ULANOWICZ, R. E. The seasonal dynamics of the chesapeake bay ecosystem. *Ecological Monographs 59* (1989), 329–364.

[4] CHAKRABARTI, D., ZHAN, Y., AND FALOUTSOS, C. R-mat: A recursive model for graph mining. In *In SDM* (2004).

[5] CHEN, L., DEVRIES, A. L., AND CHENG, C. H. Convergent evolution of antifreeze glycoproteins in Antarctic notothenioid fish and Arctic cod. *Proc Natl Acad Sci USA* (1997), 3817–3822.

[6] CLAUSET, G., NEWMAN, M. E. AND AND MOORE, C. Finding community structure in very large networks. *Physical Review E*, (2004), 1–6.

[7] GIRVAN, M., AND NEWMAN, M. E. Community structure in social and biological networks. *Proc Natl Acad Sci U S A 99*, 12 (June 2002), 7821–7826.

[8] HENDRIX, W., SCHMIDT, M. C., BREIMYER, P., AND SAMATOVA, N. F. On perturbation theory and an algorithm for maximal clique enumeration in uncertain and noisy graphs. In *U '09: Proceedings of the 1st ACM SIGKDD Workshop on Knowledge Discovery from Uncertain Data* (New York, NY, USA, 2009), ACM, pp. 48–56.

[9] HOPCROFT, J., KHAN, O., KULIS, B., AND SELMAN, B. Tracking evolving communities in large linked networks. *PNAS 101* (2004), 5249–5253.

[10] KUMAR, R., NOVAK, J., AND TOMKINS, A. Structure and evolution of online social networks. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (2006), ACM, p. 617.

[11] LONG, M., BETRAN, E., THORNTON, K., AND WANG, W. The origin of new genes: glimpses from the young and old. *Nat Rev Genet* (2003), 865–875.

[12] PALLA, G., DERENYI, I., FARKAS, I., AND VICSEK, T. Uncovering the overlapping community structure of complex networks in nature and society. *Nature 435*, 7043 (June 2005), 814–818.

[13] PALLA, G., LSZL BARABSI, A., VICSEK, T., AND HUNGARY, B. Quantifying social group evolution. *Nature 446* (2007), 2007.

[14] SCHMIDT, M. C., SAMATOVA, N. F., THOMAS, K., AND PARK, B.-H. A scalable, parallel algorithm for maximal clique enumeration. *J. Parallel Distrib. Comput. 69*, 4 (2009), 417–428.

[15] SNEL, B., BORK, P., AND HUYNEN Genome evolution. Gene fusion versus gene fission. *Trends Genet* (January 2000), 9–11.

[16] STEINHAEUSER, K., CHAWLA, N. V., AND GANGULY, A. R. An exploration of climate data using complex networks. In *SensorKDD '09: Proceedings of the Third International Workshop on Knowledge Discovery from Sensor Data* (New York, NY, USA, 2009), ACM, pp. 23–31.

[17] TANTIPATHANANANDH, C., WOLF, T. B., AND KEMPE, D. 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* (2007), ACM, pp. 717–726.

[18] WATTS, D. J., AND STROGATZ, S. H. Collective dynamics of 'small-world' networks. *Nature 393*, 6684 (1998), 440–442.

[19] ZHANG J. Evolution by gene duplication: an update. *Trends in Ecology & Evolution 18* (2003), 292–298.