Should I Stay or Should I Go: Predicting Changes in Cluster Membership

Evangelia Tsoukanara¹, Georgia Koloniari¹, and Evaggelia Pitoura²

¹ Department of Applied Informatics, University of Macedonia, Thessaloniki, Greece
{etsoukanara, gkoloniari}@uom.edu.gr

² Computer Science & Engineering, University of Ioannina, Ioannina, Greece
pitoura@cse.uoi.gr

Abstract. Most research on predicting community evolution focuses on changes in the states of communities. Instead, we focus on individual nodes and define the novel problem of predicting whether a specific node stays in the same cluster, moves to another cluster or drops out of the network. We explore variations of the problem and propose appropriate classification features based on local and global node measures. Motivated by the prevalence of machine learning approaches based on embeddings, we also introduce efficiently computed distance-based features using appropriate node embeddings. In addition, we consider chains of features to capture the history of the nodes. Our experimental results depict the complexity of the different formulations of the problem and the suitability of the selected features and chain lengths.

Keywords: cluster evolution · embeddings · feature selection · classification

1 Introduction

For the problem of community evolution, communities are monitored across time and their properties studied to attain useful conclusions. Such conclusions can then be exploited so as to predict community changes through time. Most works model community evolution through a predefined set of events, such as community growth or shrinkage, community merging or splitting and so on [5,7,12,16]. The problem is then modeled as a classification problem, where given a community history, the next event in its evolution is predicted.

However, in many applications, it is important not only to predict the behavior of a community but also of individual community members. Let us consider customers that are connected via the common products they buy. These customers can be clustered according to the companies and products they prefer. Focusing on individuals in such clusters, makes it possible for companies to identify and reward loyal customers (community members) or take preemptive measures to change the behavior of the ones that seem less dedicated.

To this end, we introduce a novel community evolution prediction problem defined at node level. For each individual node, there are three possible events:
the node may (a) stay in the same cluster, (b) move to a different cluster, or (c) drop out of the network. We study variations of the problem by considering combinations and subsets of the possible evolution events, and evaluate the use of different classification features for solving them. Firstly, we consider classic features based on popular node measures. Motivated by the advances of machine learning methods for community detection that use node embeddings, we also propose features based on distances between such embeddings. In particular, we deploy the ComE \[4\] approach that provides both node and community embeddings. Features of both methods, defined at cluster and out of cluster or network level, are combined into chains, modeling the evolution history of the nodes through time \[16\]. Our experimental results show that the problems we defined are not trivial, and that the proposed distance-based embeddings perform almost as well as the classic ones, while being computed much more efficiently.

The rest of the paper is structured as follows. Section 2 briefly describes related work. In Section 3, we formulate the novel problem and its variations. Section 4 presents our classification features and their modeling into chains. Section 5 includes our experimental results, while Section 6 concludes.

2 Related Work

We discuss related research, first, on community evolution and then, on node and graph embeddings.

Community Evolution. After discovering communities in evolving networks \[15\], their properties are studied by mapping corresponding communities through successive network snapshots \[1\]. Community evolution is assessed with measures such as its growth and disappearance rate \[20\], or its life expectancy \[10\].

For predicting community evolution \[5,12,16\], events such as community growth, shrinkage, merging and splitting are defined. The problem is modeled as a classification problem in which features based on the structural properties of communities are exploited, and given the history of a community the next event in the community’s evolution is predicted. In our work, instead of communities, we focus on nodes, and introduce a new problem aiming at predicting changes in the nodes’ cluster membership through time.

Node Embeddings. An embedding is the transformation of a high-dimensional space to a low-dimension vector. For graphs, the focus is mostly on node embeddings that preserve network structure. Deepwalk \[14\] learns node embeddings that capture second-order proximity (i.e., proximity between shared neighbors) by simulating short random walks and applying the Skip-gram algorithm. LINE \[17\] employs edge-sampling, and to preserve both first-order (i.e., ties between neighbors) and second-order proximity, it is first trained separately and then the two resulting embeddings are concatenated. Node2vec \[6\] extends Deepwalk by generating biased random walks to explore diverse node neighborhoods. Finally, GraRep \[3\] and HOPE \[9\] derive embeddings that capture high-order proximity.

Besides, link prediction, node classification and visualization, node and graph embeddings are also used for community detection \[8,19\]. ComE (Community
Embedding) [4] is a framework that jointly solves both community detection, and learning node and community embeddings. The intuition is that node embeddings that capture community-aware proximity, can assist community detection, while community embeddings can in turn improve node embeddings. In our work, we deploy ComE and explore whether node and community embeddings can be used to derive predictive features.

3 Problem Formulation

A social network is often represented as a graph \( G = (V, E) \), where \( V \) is the set of nodes (vertices) and \( E \) is the set of edges. A temporal social network is a network that changes over time and is represented as a sequence of graphs \( \{G_1, G_2, \ldots, G_n\} \), where \( G_i = (V_i, E_i) \), represents a snapshot of graph \( G \) at time \( i \), and \( V_i \) and \( E_i \) are the node and edge sets at time \( i \) respectively. Let \( C_i = \{C_1^i, C_2^i, \ldots, C_m^i\} \) be a clustering of \( G_i \) consisting of \( m \) clusters, such that \( C_j^i \cap C_k^i = \emptyset \), \( 1 \leq j, k \leq m, j \neq k \). For two clusterings \( C_i \) and \( C_{i+1} \) at consecutive timeframes \( i \) and \( i+1 \), we assume cluster \( C_{i+1}^j \) is the evolution of cluster \( C_i^j \). Similarly, for multiple consecutive timeframes, \( C_1^j, C_2^j, \ldots, C_n^j \) denotes the evolution of cluster \( C_j^i \) in time period \([1, n]\).

Inspired by the idea of community evolution, we study the problem at node level by aiming to predict how node memberships in clusters evolve through time. We discern between different states that a node can have with respect to its cluster membership in the next timeframe, i.e., a node can stay in the same cluster, move to another or drop out of the network. Based on the above, we define our problem as follows.

**Definition 1 (Stay\Move\Drop (SMD) Problem).** Given a sequence of clusterings \( C_1, \ldots, C_i \), corresponding to a consecutive set of timeframes for a graph \( G \), and node \( v \in C_j^i \), predict the state of node \( v \) regarding its evolution in the next timeframe \( i + 1 \) as state:
- **stay**, \( S \): node \( v \) stays at the same cluster in \( i + 1 \), that is \( v \in C_{i+1}^j \),
- **move**, \( M \): node \( v \) moves to another cluster in \( i + 1 \), that is \( v \in C_k^{i+1} \), \( k \neq j \), and
- **drop**, \( D \): node \( v \) drops out of the network, that is \( v \notin V_{i+1} \).

Thus, we define a classification problem with 3 classes, labeled stay, move and drop. Given the history of a node in a given time period, defined as a sequence of distinct timeframes, we predict its class in the next time frame.

If we are only interested in discerning between loyal cluster members and members likely to leave, we may simplify our problem to a binary classification problem. This first alternative problem, is derived by merging states move and drop, in one class leave. Thus, the classes are reformed as follows:

**Stay\Leave (SL) Problem:** the possible node events are reformed as state:
- **stay**, \( S \): node \( v \) stays at the same cluster in \( i + 1 \), that is \( v \in C_{i+1}^j \), and
- **leave**, \( L \): node \( v \) does not remain in the same cluster, that is \( v \notin C_j^{i+1} \).
Finally, we omit the third class of the SMD problem, and only provide predictions for nodes that remain in the network in timeframe \( i + 1 \). For the third variation, we have:

**Stay\Move (SM) Problem.**

- **stay, **\( S \): node \( v \) stays at the same cluster in \( i + 1 \), that is \( v \in C^j_{i+1} \), and
- **move, **\( M \): node \( v \) moves to another cluster in \( i + 1 \), that is \( v \in C^k_{i+1}, k \neq j \).

![Fig. 1: (a) ComE clustering and \( \Delta \) distribution for (b) Syntgen and (c) DBLP.](image)

### 4 Predictive Features & Historical Chains

To solve the classification problems we define, the evaluation and selection of appropriate predictive features is required. We discern between two basic types of features, based on structural network measures that are usually exploited for community evolution predictions, and on distances between embeddings that we propose. As all three problems we introduce are variations of the same classification problem, we define and evaluate the same features for all problems.

**Classic Features.** Classic features, defined on node level, provide information about the structural role of a node in the network. In particular, we select:

- **degree** measuring the connections of a node,
- **betweenness**, measuring the number of shortest paths that pass through a node,
- **closeness**, that measures the distance of a node to all other nodes, and
- **eigenvector centrality** measuring the influence of a node in the network, defined on cluster and network level.

Aggregated at community level, classic features offer insights for predicting community evolution \[^{12,16}\], while for individual nodes, such measures also contain information about their evolution tendencies. For instance, an influential central node with high degree is less likely to drop out of the network compared to a low-degree remote node. Similarly focusing on community structure, well-connected core nodes within a community are more likely to stay in their cluster compared to loosely connected border nodes. Thus, we also differentiate between features defined at cluster level (\( \text{in} \)) and at network level (\( \text{out} \)).

**Embeddings-based Features.** Since node embeddings are low dimensional vector representations of nodes that capture structural graph properties, we propose defining predictive features based on such embeddings.
framework provides an appropriate solution by solving both community detection and embeddings learning jointly, focusing on deriving embeddings that capture community-aware proximity between nodes. ComE uses as input a graph’s edge list and the number of target clusters $k$ and outputs: (i) for each node its embedding along with its community membership and (ii) for each community, which is defined as a multivariate Gaussian distribution, its embedding parameters, that is, a median vector (i.e., the embedding of the mean of the community) and a covariance matrix. Fig. 1a depicts the node embeddings for the six clusters detected by ComE for a synthetic dataset generated by the Syntgen [13] generator for timeframe $T_0$, where we notice that ComE manages to detect reasonable well-separated clusters with similar node embeddings as projected in 2-d.

We define features on cluster and network level, exploiting the outputs of ComE. For computing network level features, we exclude the nodes of the cluster the given node belongs to, to avoid cases of identical network and cluster features. Let $\phi_v$ be the embedding of node $v$, and $\phi_C$ the embedding of the median of community $C$. Without loss of generality we define our features using the euclidean distance ($d$) between pairs of embeddings. Cosine and $L_1$ distances were also evaluated, but euclidean were used as they performed slightly better. In particular, for a node $v$ such that $v \in C$, we define the following features:

**Cluster level:**
- distance from cluster median: $d(\phi_v, \phi_C)$
- distance from least similar cluster member: $\max_{\forall u \in C} d(\phi_v, \phi_u)$
- distance from most similar cluster member: $\min_{\forall u \in C} d(\phi_v, \phi_u)$
- avg. distance from all cluster members: $\frac{1}{|C|} \sum_{\forall u \in C} d(\phi_v, \phi_u)$

**Network level:**
- min. distance from other cluster median: $\min_{\forall C' \neq C} d(\phi_v, \phi_{C'})$
- distance from least similar node out of the cluster: $\max_{\forall u \not\in C} d(\phi_v, \phi_u)$
- distance from most similar node out of the cluster: $\min_{\forall u \not\in C} d(\phi_v, \phi_u)$
- avg. distance from all nodes not in the cluster: $\frac{1}{|V| - |C|} \sum_{\forall u \not\in C} d(\phi_v, \phi_u)$

Our approach is based on the idea that the embeddings of nodes in a cluster are more similar. Thus, if a node has an embedding that is similar to nodes of other clusters, it is more likely to change or leave its cluster. In Fig. 1b and Fig. 1c, we plot the distribution of the difference, $\Delta$, between $\min_{\forall C' \neq C} d(\phi_v, \phi_{C'})$ and $d(\phi_v, \phi_C)$ features, for classes stay and move for a synthetic dataset generated by Syntgen and a citation DBLP dataset based on [18]. In both figures, we notice that nodes that move to another cluster tend to have lower or even negative $\Delta$ compared to the ones that remain in the same cluster. About 60% of the move nodes have $\Delta$ less than 0.20 for both datasets, while more than 60% of the stay nodes have $\Delta$ higher than 0.28. Therefore, the values of the various distance measures can provide insight on the properties and behavior of a node, and thus, are appropriate as predictive features for our classification problems.

**Historical Chains of Features.** The evolution of a community is tracked in successive network snapshots that correspond to successive timeframes. Thus, all features we describe can be measured for each timeframe. Let us assume a
set of \( k \) predictive features for each node \( v \), and a time period \([1, \ldots, n]\), where \( f_i^j(v) \) denotes the \( j \)-th feature of node \( v \) at time \( i \). Further, for every pair of consecutive timeframes \( i \) and \( i + 1 \) in the given time period, the state (label), \( l_{i+1}(v) \), of node \( v \) can be recorded.

To model the history of the node and exploit it to derive more accurate predictions, we utilize historical chains of features as defined in [16]. In particular, we have as final features for \( v \):

\[
\{f_1^1(v), \ldots, f_k^1(v)\}, \{f_2^2(v), \ldots, f_k^2(v)\}, l_3(v) \ldots, \{f_1^n(v), \ldots, f_k^n(v)\}, \text{while } l_{n+1}(v) \text{ is the label to be predicted}.
\]

Though we have defined here one chain to model the entire node history, the use of subchains of various lengths can also be deployed. While a longer history will provide more information regarding the history of a node, it would limit the number of nodes for which a prediction can be made.

Table 1: Snapshot Structure

| Sn | DBLP | Email-eu | Syntgen |
|----|------|----------|---------|
|    | Nodes Edges | | Nodes Edges | | Nodes Edges | |
| 0  | 14731 120192 | 17 | 0.681 | 750 4740 | 11 | 0.515 | 1583 8955 | 6 | 0.525 |
| 1  | 16901 143404 | 16 | 0.676 | 745 5077 | 11 | 0.431 | 1443 7936 | 5 | 0.480 |
| 2  | 17756 156393 | 16 | 0.649 | 742 4578 | 10 | 0.528 | 1367 7249 | 5 | 0.459 |
| 3  | 14765 120370 | 17 | 0.659 | 742 4886 | 9 | 0.487 | 1567 8154 | 5 | 0.476 |
| 4  | 10898 69005 | 16 | 0.660 | 749 5072 | 11 | 0.388 | 1575 7993 | 5 | 0.483 |
| 5  | 739 4819 | 10 | 0.521 | 739 4819 | 10 | 0.521 | 1402 6978 | 5 | 0.465 |
| 6  | 759 4846 | 10 | 0.521 | 759 4846 | 10 | 0.521 | 1526 7629 | 5 | 0.472 |
| 7  | 808 5405 | 11 | 0.410 | 808 5405 | 11 | 0.410 | 1416 6973 | 5 | 0.462 |
| 8  | 772 4890 | 13 | 0.391 | 772 4890 | 13 | 0.391 | 1409 6853 | 5 | 0.470 |
| 9  | 785 5169 | 12 | 0.533 | 785 5169 | 12 | 0.533 | 1594 7794 | 6 | 0.490 |

5 Evaluation

We experimentally study all three classification problems while comparing different sets of predictive features.

5.1 Datasets

We use three datasets for our evaluation, two real and one synthetic.

**DBLP**: DBLP is a citation network\(^3\) that includes additional information about the publications, such as year of publication and fields of study they belong to [18]. Similarly to [4], we filter papers with primary of study as NLP, Databases, Networking, Data Mining and Computer Vision. We construct five snapshots, for years 2015 to 2019, by maintaining publications of the given year and adding cited papers that belong to the selected fields regardless of their publication year. To attain a denser network, we sample nodes with degree \( \geq 20 \) and build the induced undirected subgraph.

\(^3\)https://www.aminer.org/citation
Table 2: Performance for the SM Problem

| Data  | Feat. | Class | ComE | Classic |
|-------|-------|-------|------|---------|
|       |       |       | P    | R      | F1    | Acc  | P    | R      | F1    | Acc  |
|       |       |       |      |        |       |      |      |        |       |      |
| DBLP  | in    | S     | 0.821| 0.940 | 0.876 | 0.804| 0.837| 0.941 | 0.886 | 0.821|
|       |       | M     | 0.715| 0.425 | 0.533 | 0.746| 0.485| 0.587 |      |      |
|       | out   | S     | 0.800| 0.942 | 0.865 | 0.784| 0.783| 0.927 | 0.849 | 0.757|
|       |       | M     | 0.679| 0.339 | 0.452 | 0.577| 0.279| 0.376 |      |      |
|       | all   | S     | 0.836| 0.954 | 0.891 | 0.828| 0.839| 0.948 | 0.890 | 0.827|
|       |       | M     | 0.787| 0.476 | 0.592 | 0.769| 0.490| 0.599 |      |      |
| Email-eu | in | S     | 0.807| 0.926 | 0.862 | 0.796| 0.797| 0.912 | 0.850 | 0.778|
|       |       | M     | 0.757| 0.597 | 0.666 | 0.712| 0.483| 0.574 |      |      |
|       | out   | S     | 0.778| 0.933 | 0.848 | 0.770| 0.732| 0.910 | 0.812 | 0.709|
|       |       | M     | 0.733| 0.409 | 0.524 | 0.569| 0.281| 0.375 |      |      |
|       | all   | S     | 0.825| 0.954 | 0.875 | 0.816| 0.789| 0.925 | 0.852 | 0.778|
|       |       | M     | 0.792| 0.554 | 0.652 | 0.731| 0.452| 0.558 |      |      |
| Syntgen | in    | S     | 0.675| 0.692 | 0.683 | 0.668| 0.707| 0.719 | 0.713 | 0.700|
|       |       | M     | 0.661| 0.643 | 0.652 | 0.693| 0.680| 0.687 |      |      |
|       | out   | S     | 0.638| 0.704 | 0.670 | 0.649| 0.642| 0.669 | 0.655 | 0.635|
|       |       | M     | 0.643| 0.572 | 0.606 | 0.628| 0.599| 0.613 |      |      |
|       | all   | S     | 0.693| 0.725 | 0.708 | 0.694| 0.710| 0.713 | 0.711 | 0.701|
|       |       | M     | 0.690| 0.656 | 0.672 | 0.691| 0.687| 0.689 |      |      |

Table 3: Performance for the SL Problem

| Data  | Feat. | Class | ComE | Classic |
|-------|-------|-------|------|---------|
|       |       |       | P    | R      | F1    | Acc  | P    | R      | F1    | Acc  |
|       |       |       |      |        |       |      |      |        |       |      |
| DBLP  | in    | S     | 0.626| 0.670 | 0.650 | 0.922| 0.715| 0.687 | 0.700 | 0.937|
|       |       | L     | 0.961| 0.951 | 0.956 | 0.962| 0.967| 0.965 | 0.932| 0.932|
|       | out   | S     | 0.632| 0.664 | 0.647 | 0.922| 0.680| 0.648 | 0.664 | 0.929|
|       |       | L     | 0.959| 0.953 | 0.956 | 0.958| 0.963| 0.960 | 0.932| 0.932|
|       | all   | S     | 0.680| 0.710 | 0.693 | 0.933| 0.728| 0.690 | 0.709 | 0.939|
|       |       | L     | 0.965| 0.960 | 0.962 | 0.963| 0.969| 0.966 | 0.932| 0.932|
| Email-eu | in | S     | 0.796| 0.923 | 0.855 | 0.820| 0.799| 0.892 | 0.843 | 0.809|
|       |       | L     | 0.869| 0.880 | 0.872 | 0.827| 0.698| 0.756 | 0.809| 0.809|
|       | out   | S     | 0.771| 0.915 | 0.837 | 0.795| 0.725| 0.882 | 0.796 | 0.740|
|       |       | L     | 0.847| 0.638 | 0.724 | 0.775| 0.548| 0.642 | 0.740| 0.740|
|       | all   | S     | 0.809| 0.934 | 0.867 | 0.836| 0.793| 0.892 | 0.840 | 0.804|
|       |       | L     | 0.889| 0.703 | 0.758 | 0.826| 0.686| 0.749 | 0.804| 0.804|
| Syntgen | in    | S     | 0.656| 0.654 | 0.655 | 0.708| 0.696| 0.690 | 0.693 | 0.741|
|       |       | L     | 0.747| 0.748 | 0.747 | 0.774| 0.778| 0.776 | 0.741| 0.741|
|       | out   | S     | 0.634| 0.660 | 0.646 | 0.694| 0.631| 0.625 | 0.628 | 0.686|
|       |       | L     | 0.742| 0.720 | 0.731 | 0.726| 0.731| 0.729 | 0.686| 0.686|
|       | all   | S     | 0.670| 0.682 | 0.676 | 0.723| 0.693| 0.684 | 0.688 | 0.738|
|       |       | L     | 0.763| 0.753 | 0.758 | 0.770| 0.778| 0.774 | 0.738| 0.738|
**Email-eu:** The Email-eu dataset consists of incoming and outgoing e-mails between members of a large European institution in a period of 803 days. We consider the network undirected and split the data into 10 balanced snapshots.

**Syntgen:** To further investigate the impact of network properties on our problems, we use the Syntgen generator that creates temporal undirected networks simulating real networks using explicit specifications, like degree distributions and cluster sizes, as well as implicitly controlling the perseverance of nodes popularity over time. The intra-cluster to total degree ratio determines cluster density. A high ratio leads to dense well-separated communities, while low values exhibit no clustering. We set the default ratio to 0.7.

We apply ComE to detect communities at each snapshot. To determine an appropriate number of clusters \( k \) as input for ComE, for DBLP and Email-eu, we first apply the Louvain community detection method that selects the \( k \) that maximizes modularity. As to Syntgen, we use as \( k \) the number of clusters obtained from the generator. Clusters are mapped across different snapshots based on the majority of their common nodes. Table 1 presents the number of nodes and edges as well as the number of clusters \( |C| \) and modularity \( Q \) for each network snapshot for all datasets.

With regards to historical chains, DBLP with 5 snapshots can form 2-length up to 4-length chains, while Email-eu and Syntgen with 10 snapshots can form from 2-length up to 9-length chains. Trying to balance between more information that longer chains provide and the ability to provide predictions for more nodes, we use as default chain length 5 for both Email-eu and Syntgen, and 2 for DBLP.

![Graphs](image)

(a) \( SM \) Problem  
(b) \( SL \) Problem  
(c) \( SMD \) Problem

**Fig. 2:** Macro f1 score per class.

### 5.2 Experimental Results

We report results using the Random Forest classifier, which performed better compared to other classifiers we tried (e.g. Logistic Regression, Naive Bayes). We apply stratified 5-fold cross-validation to preserve the same class distribution in both train and test sets. Tables 2 to 4 illustrate precision, recall and f1-score per class, and accuracy for each problem when using ComE and Classic features at cluster (in) and network level (out), and their combination (all) for each dataset. We denote with bold the best f1-score for each class per dataset and problem.

---

4 https://snap.stanford.edu/data/email-Eu-core-temporal.html
As structural features are typically used in community evolution prediction, we consider Classic features as a baseline method to compare the proposed ComE-based features.

**General observations.** A first observation derived by Table 2 to Table 4 is that while the three classification problems are not trivial, both Classic and ComE based features show promising initial results, achieving high accuracy but showcasing that some classes are more difficult to predict than others. We also notice that all features perform better in most cases, and use them as our default features for the rest of our study.

### Table 4: Performance for the $SMD$ Problem

| Data  | Feat. | Class | P     | R     | F1    | Acc   | P     | R     | F1    | Acc   |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
|       |       |       | ComE  |       |       |       | Classic |       |       |       |
|       |       | $in$  |       |       |       |       |       |       |       |       |
|       |       | $S$   | 0.589 | 0.818 | 0.685 | 0.666 | 0.778 | 0.718 |
|       |       | $M$   | 0.538 | 0.358 | 0.430 | 0.587 | 0.315 | 0.410 | 0.917 |
|       |       | $D$   | 0.973 | 0.941 | 0.957 | 0.962 | 0.962 | 0.962 |
|       |       | $out$ |       |       |       |       |       |       |       |       |
|       |       | $S$   | 0.621 | 0.853 | 0.718 | 0.682 | 0.778 | 0.727 |
|       |       | $M$   | 0.601 | 0.385 | 0.468 | 0.616 | 0.310 | 0.410 | 0.911 |
|       |       | $D$   | 0.972 | 0.944 | 0.958 | 0.961 | 0.966 | 0.963 |
|       |       | $all$ |       |       |       |       |       |       |       |       |
|       |       | $S$   | 0.621 | 0.853 | 0.718 | 0.682 | 0.778 | 0.727 |
|       |       | $M$   | 0.601 | 0.385 | 0.468 | 0.616 | 0.310 | 0.410 | 0.911 |
|       |       | $D$   | 0.972 | 0.944 | 0.958 | 0.961 | 0.966 | 0.963 |
| Email-eu |      | $in$  |       |       |       |       |       |       |       |       |
|       |       | $S$   | 0.795 | 0.932 | 0.858 | 0.783 | 0.909 | 0.841 |
|       |       | $M$   | 0.748 | 0.517 | 0.611 | 0.653 | 0.468 | 0.545 |
|       |       | $D$   | 1     | 0.881 | 0.936 | 0.996 | 0.881 | 0.935 |
|       |       | $out$ |       |       |       |       |       |       |       |       |
|       |       | $S$   | 0.762 | 0.927 | 0.836 | 0.718 | 0.809 | 0.798 |
|       |       | $M$   | 0.686 | 0.413 | 0.515 | 0.524 | 0.272 | 0.357 |
|       |       | $D$   | 1     | 0.881 | 0.936 | 0.993 | 0.881 | 0.933 |
|       |       | $all$ |       |       |       |       |       |       |       |       |
|       |       | $S$   | 0.797 | 0.939 | 0.862 | 0.775 | 0.905 | 0.835 |
|       |       | $M$   | 0.765 | 0.520 | 0.619 | 0.647 | 0.451 | 0.530 |
|       |       | $D$   | 1     | 0.881 | 0.937 | 0.996 | 0.881 | 0.932 |
| Syntgen |      | $in$  |       |       |       |       |       |       |       |       |
|       |       | $S$   | 0.645 | 0.680 | 0.662 | 0.683 | 0.707 | 0.695 |
|       |       | $M$   | 0.634 | 0.646 | 0.640 | 0.668 | 0.696 | 0.681 |
|       |       | $D$   | 1     | 0.828 | 0.906 | 1     | 0.828 | 0.905 |
|       |       | $out$ |       |       |       |       |       |       |       |       |
|       |       | $S$   | 0.620 | 0.692 | 0.654 | 0.620 | 0.658 | 0.638 |
|       |       | $M$   | 0.628 | 0.599 | 0.631 | 0.663 | 0.610 | 0.606 |
|       |       | $D$   | 1     | 0.828 | 0.906 | 1     | 0.828 | 0.906 |
|       |       | $all$ |       |       |       |       |       |       |       |       |
|       |       | $S$   | 0.661 | 0.712 | 0.685 | 0.685 | 0.706 | 0.695 |
|       |       | $M$   | 0.659 | 0.657 | 0.658 | 0.668 | 0.698 | 0.682 |
|       |       | $D$   | 1     | 0.828 | 0.906 | 1     | 0.828 | 0.906 |

$SM$ vs. $SL$ vs. $SMD$. To compare the three problems, we illustrate the f1-score for each class for both types of features (ComE, denoted as CM, and Classic denoted as CL) for the three datasets in Fig. 2. For the $SM$ problem, Fig. 2a shows that class stay performs better than move for DBLP and Email-eu. DBLP
achieves the best performance with 0.891 for stay and 0.592 for move respectively for the ComE features and similar results for the Classic ones (Table 2). This is due to the imbalance in the real datasets between the two classes, with the majority of the nodes in class stay. In contrast, in the Syntgen dataset, classes are well-balanced and we notice similar performance for both. For the SL problem (Fig. 2b), we notice significant difference mainly on DBLP. In this case, class stay is underrepresented due to the network construction. Looking at Table 3, class stay achieves f1 0.695 and leave 0.962, for DBLP with feature type all. Finally, as we can see in Table 4 and Fig. 2c for the SMD problem, class move is heavily underrepresented in both DBLP and Email-eu resulting in a rather low f1-score. Summing up, the SL problem seems to have the best overall performance, while SM appears to have the worst. Apparently, class move is overall the most difficult to predict. Besides class imbalance that makes the problem more difficult, this occurs especially when the characteristics across communities are not significantly different, which is a similarity indicator between communities.

Selecting appropriate features. Next, we focus on the comparison between the different types of features in (I), out (O) and all (A) for ComE (CM) and Classic (CL) features and how they perform at each problem. As we can see in Fig. 3, in and all outperform out features. For ComE features all is the best choice for all problems, while for Classic features in performs sometimes better. This seems to depend on the dataset and not the problem we study, as we see that for Email-eu Classic in features perform better for all problems (with the exception of class stay on the SM problem). For the SM problem and ComE features, we notice a significant difference between out and all features (Fig. 3a). Macro f1 is 0.613 for out and 0.744 for all for the DBLP dataset and 0.584 for out and 0.704 for all for the Email-eu dataset. Overall, we do not observe significant differences between ComE and Classic features, deducing that the ComE features offer satisfying performance while being more efficiently computed. For instance, for the DBLP dataset, the computation time is 5493s for ComE features, while Classic features are slower by an order, requiring 34618s.

Influence of the length of historical chains. Depicted in Fig. 4a, 4b, and 4c we explore the effect of chains of features with varying length for the DBLP (D), Email-eu (E) and Syntgen (S). Both Email-eu and Syntgen show that f1
generally improves as chain length grows for all problems. The Syntgen dataset follows the same pattern for the SL and SMD problems, increasing and reaching its peak at chain length 8. For the Email-eu, we observe a temporary drop at length 7, while the highest score is reached at length 9, with 0.804 for the SM (Fig. 4a), 0.883 for the SL (Fig. 4b) and 0.850 for the SMD (Fig. 4c) problem respectively. As we have mentioned, while longer chains provide more information and more accurate predictions, they are not available for a large number of nodes. In particular, Email-eu consists of 6034 instances at 2-length chain and only 750 instances at its longest chain. Similarly, Syntgen consists of 11879 instances at 2-length and 1583 instances at 9-length chain. With regards to DBLP, the best score, 0.828, is achieved at the SL problem with chain length 2 with 0.828, but the history is too limited to derive safe comparative conclusions.

Influence of intra-cluster to total degree ratio. In the last experiment, we focus on the Syntgen generator producing different datasets with varying intra-cluster to total degree ratio, which determines the density within the constructed clusters compared to the overall network. In Fig. 4d, we notice a sharp increase on macro f1 for ratio 0.6 up to 0.8 for all problems. Lower ratio indicates poor clustering and thus is not suitable for our context. Beyond 0.8, behavior diverges. In such tightly connected communities, most nodes have similar roles in their community, making it difficult for a classifier to determine their behavior. As a conclusion, cases with ratio close to 0.5 that exhibit no locality, or close to 0.9 indicating almost disconnected communities, fail to achieve good results.

6 Conclusions

In this paper, we defined a novel problem, related to community evolution, that focuses on nodes and aims at predicting whether they will stay in their cluster, move to another or drop out of the network. We modeled the problem as a classification problem and with three variations. We determined appropriate features, based on both local and global node measures, and formed chains of features to take advantage of node history. We also proposed exploiting node and community embeddings derived by the ComE framework to define distance based features. Our experimental results showed that the novel problem is not trivial, and the distance-based features performed similarly to the Classic ones, while requiring far less computation time. Next, we will consider alternative community learning approaches to derive node embeddings and define appropriate features.
References

1. Aynaud, T., Fleury, E., Guillaume, J.L., Wang, Q.: Communities in evolving networks: Definitions, detection, and analysis techniques. In: Dynamics On and Of Complex Networks, Volume 2: Applications to Time-Varying Dynamical Systems, pp. 159–200. Springer New York (2013)
2. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment 2008(10), P10008 (2008)
3. Cao, S., Lu, W., Xu, Q.: Grarep: Learning graph representations with global structural information. In: Proc. of the 24th ACM CIKM. p. 891–900 (2015)
4. Cavallari, S., Zheng, V.W., Cai, H., Chang, K.C.C., Cambria, E.: Learning community embedding with community detection and node embedding on graphs. In: Proc. of the 2017 ACM CIKM. p. 377–386 (2017)
5. Gliwa, B., Bródká, P., Zygmunt, A., Saganowski, S., Kazienko, P., Kożłak, J.: Different approaches to community evolution prediction in blogosphere. In: Proc. of the 2013 IEEE/ACM ASONAM. p. 1291–1298 (2013)
6. Grover, A., Leskovec, J.: Node2vec: Scalable feature learning for networks. In: Proc. of the 22nd ACM SIGKDD. p. 855–864 (2016)
7. Ilhan, N., Oğüdücü, Ş.G.: Predicting community evolution based on time series modeling. In: Proc. of the 2015 IEEE/ACM ASONAM. p. 1509–1516 (2015)
8. Kozdoba, M., Mannor, S.: Community detection via measure space embedding. In: Advances in Neural Inf. Proc. Sys. 28: NIPS 2015. pp. 2890–2898 (2015)
9. Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W.: Asymmetric transitivity preserving graph embedding. In: Proc. of the 22nd ACM SIGKDD. p. 1105–1114 (2016)
10. Palla, G., Barabási, A.L., Vicsek, T.: Quantifying social group evolution. Nature 446, 664–667 (2007)
11. Paranjape, A., Benson, A.R., Leskovec, J.: Motifs in temporal networks. In: Proc. of the Tenth ACM WSDM. p. 601–610 (2017)
12. Pavlopoulou, M.E.G., Tzortzis, G., Vogiatzis, D., Paliouras, G.: Predicting the evolution of communities in social networks using structural and temporal features. In: 12th Int. Work. on Semantic and Social Media Adaptation and Personalization. pp. 40–45 (2017)
13. Pereira, L.R., Lopes, R.J., Louçã, J.: Syntgen: a system to generate temporal networks with user-specified topology. Journal of Complex Networks 4(0), 1–26 (2019)
14. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proc. of the 20th ACM SIGKDD. p. 701–710 (2014)
15. Rossetti, G., Cazabet, R.: Community discovery in dynamic networks: A survey. ACM Comput. Surv. 51(2) (2018)
16. Saganowski, S.: Predicting community evolution in social networks. In: Proc. of the 2015 IEEE/ACM ASONAM. p. 924–925 (2015)
17. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: Proc. of the 24th WWW. p. 1067–1077 (2015)
18. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: Arnetminer: Extraction and mining of academic social networks. In: Proc. of the 14th ACM SIGKDD. p. 990–998 (2008)
19. Tian, F., Gao, B., Cui, Q., Chen, E., Liu, T.Y.: Learning deep representations for graph clustering. In: Proc. of the 28th AAAI. p. 1293–1299 (2014)
20. Toyoda, M., Kitsuregawa, M.: Extracting evolution of web communities from a series of web archives. In: Proc. of the 14th ACM Conf. on Hypertext and Hypermedia. p. 28–37 (2003)