Semi-supervised Community Detection in Dynamic Graphs

Matteo Bianco, Luca Cagliero* and Luca Vassio

Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10129 Torino TO, Italy

Abstract

Semi-supervised approaches to Community Detection (CD) in graphs aim to detect communities closely related to a few labeled ones. State-of-the-art semi-supervised algorithms adopt a three-step process, which entails (1) Generating candidate communities based solely on the network structure; (2) Selecting the candidates that are most similar to the labeled communities; (3) Refining the selected communities shortlisted at Step (2). However, existing approaches are unsuited to handle the dynamics in labeled communities and their relations with time-varying graph structures. In this work, we formulate the new task of semi-supervised CD from dynamic graphs, which is relevant to real-world time-evolving scenarios. To avoid executing the previous pipeline independently at every time step and potentially missing relevant temporal community-level relations, we envisage a new approach relying on time-aware strategies for both dynamic graph embedding and community selection and refinement. We leverage a latent graph representation incorporating node- and subgraph-level temporal relations neglected by static approaches. Then, supervised community refinements are propagated across consecutive time steps to capture time-evolving trends. After adapting static CD models to the dynamic scenario, we conduct extensive comparisons of the methods on datasets with varying characteristics in the novel task of dynamic semi-supervised CD. The proposed approach shows remarkable improvements in low-modularity and low-stability dynamic graphs.

Keywords

Community Detection, Dynamic Graphs, Semi-Supervised Learning

1. Introduction

Communities in graphs are groups of nodes with distinctive features or connections [1]. Automating the process of Community Detection (CD) from graphs is a well-established machine learning problem. It has found application in several fields among which social network analysis [2], scientometrics [3], and healthcare [4].

To deeply explore the network graph structure, unsupervised approaches to CD have attempted to use a variety of classical data mining or Deep Learning techniques such as clustering [5], graph mining [6], and Variational AutoEncoders [7]. However, they struggle to find communities of nodes with functional relations that are not directly derivable from the network structure [8]. To tackle this issue, Semi-Supervised CD (SS-CD) approaches leverage a few examples of labeled communities, typically provided by domain experts. To efficiently address SS-CD on large graphs, state-of-the-art approaches (e.g., [9, 10, 11]) first encode graph nodes or subgraphs using graph embedding techniques. Next, they extract candidate communities based solely on the network structure. Finally, they shortlist and refine the extracted candidates by maximizing their similarity with the labeled communities in the embedding space.

Since real-world communities are naturally time-evolving, there is an increasing need to extend existing CD solutions suited to static graphs towards dynamic scenarios. Recent approaches to CD capture time-evolving trends in dynamic graphs by learning temporal graph embeddings. They either learn temporal relations in sequences of graph snapshots [12, 13, 14] or rely on parametric distance dynamic models [15]. However, to the best of our knowledge, all prior works on CD from dynamic graphs are unsuited to a semi-supervised scenario where labeled communities change over time. This calls for new approaches addressing SS-CD from dynamic graphs, in which temporal relations between graph snapshots and time-varying labeled communities are jointly analyzed.

Contribution Firstly, we formalize the Semi-Supervised Dynamic Community Detection (SS-DCD) task (Section 2). Unlike the unsupervised task, here the CD process is guided by labeled communities. Secondly, we propose an approach to tackle the newly proposed SS-DCD task (Section 3). Unlike all existing works (e.g., [9, 10]), our approach leverages temporal graph embeddings to incrementally update the dynamic graph representation. Furthermore, the steps of supervised community selection and refinement are time-aware. The aim is to consider not only the temporal relations between network graphs in the different snapshots but also the evolution of the labeled communities. Lastly, we empirically compare the performance of our approach and baseline methods under the unexplored SS-DCD setting (Section 5).

Evaluation We adapt real static graphs with ground-truth communities and various topological characteristics to a dynamic scenario (Section 4). On top of dynamic graphs, we assess both our approach and baseline methods in terms of the established F1, Jaccard, and ONMI performance scores. The results show that our approach performs on average the best on the analyzed datasets in terms of F1, Jaccard, and ONMI scores (19 wins out of 27 combinations of datasets and metrics). The results are mainly influenced by the network modularity and the level of dynamics in the sequence of graph snapshots and corresponding labels. Specifically, when the graph has a high modularity the communities are relatively easy to detect from the current network topology regardless of its past snapshots and labeled communities. The experiments confirm that the lower the modularity the higher the benefits of the newly proposed time-aware semi-supervised strategy. Similarly, the temporal stability of the network [16] is another important indicator of the level of complexity of the SS-DCD task. Our results confirm the better suitability of the proposed approach to dynamic scenarios compared to state-of-the-art approaches.

Published in the Proceedings of the Workshops of the EDBT/ICDT 2025 Joint Conference (March 25-28, 2025), Barcelona, Spain

*Corresponding author.

matteo.bianco@studenti.polito.it (M. Bianco); luca.vassio@polito.it (L. Cagliero); luca.vassio@polito.it (L. Vassio)

0000-0002-7185-5247 (L. Cagliero); 0000-0002-2920-1856 (L. Vassio)

2. Problem statement

We first introduce the preliminary concepts and related notation and then formalize the newly proposed *Semi-Supervised Dynamic Community Detection* (SS-DCD) task.

Let G^t=(V^t,E^t) be a graph where V^t and E^t are the corresponding sets of nodes and edges, respectively. A community c^t detected from G^t is a subset of the nodes with peculiar characteristics in terms of either network graph structure or node properties. Given a graph G^t, the *Unsupervised Community Detection* (U-CD) task has the goal of extracting the set C^t of all its communities without any prior knowledge on the community properties, i.e., U-CD exclusively relies on the network graph structure. When, instead, the extraction process is guided by a given set $\hat{C}^t$ of labeled communities in G^t, the task is commonly denoted by *Supervised Community Detection* (S-CD). The key idea behind S-CD is to detect communities in graphs that are closely related to the labeled ones by learning a model that automatically captures similarities among subgraphs. Importantly, S-CD is not necessarily based on the network structure solely but might consider functional properties of nodes or edges as well [8]. In this work, we assume that the CD task is *partially* supervised, i.e., the input set of labeled communities is incomplete. Given a graph G^t and a training set $\hat{C}^t$ consisting of few labeled communities in G^t, we denote by *Semi-Supervised Community Detection* (SS-CD) the task of automatically detecting all the communities in G^t.

We aim to model the temporal variations of a graph structure and its underlying communities within a reference time period. Without loss of generality, we assume that the reference period is divided into T discrete consecutive time steps (i.e., $t \in \{1, 2, \ldots, T\}$).

Dynamic Community Detection (DCD) aims to detect all communities from sequences of graph snapshots $\mathcal{G} = \{G^t\}_{t=1}^{T}$. Unsupervised approaches to DCD extract communities from G^t for every $t \in \{1, 2, \ldots, T\}$ in the absence of labeled communities. Conversely, Supervised DCD (S-DCD) leverages the set $\hat{C}^t$ of labeled communities occurring in each time step t. However, similar to the time-invariant case, the set of labeled communities is likely incomplete due to the lack of human annotations or reliable community descriptors. To tackle the above issue, we formalize the new task of *Semi-Supervised Dynamic Community Detection* (SS-DCD, in short). The twofold aims are, on the one hand, to make SS-CD time-aware by leveraging CD semi-supervision and, on the other hand, to enrich Unsupervised DCD with a combined analysis of the properties of both the network structure and the labeled communities across different snapshots.

SS-DCD task formulation Given a sequence of graph snapshots $\mathcal{G}$ and a partial set $\hat{C}^t$ of labeled communities occurring at every time step $t \in \{1, 2, \ldots, T\}$, the SS-DCD task aims is to extract the sequence $\mathcal{C} = \{C^t\}_{t=1}^{T}$ of community sets $C^1, \ldots, C^T$.

3. Semi-supervised Community Detection from Dynamic Graphs

The classical SS-CD pipeline entails the following steps:

1. *Candidate community generation*, aimed to extract candidate communities based on the structure of the network graph.
2. *Supervised candidate selection*, which reduces the set of candidates generated at the previous step according to their similarity with the labeled communities.
3. *Community refinement*, aimed to revise the generated communities, e.g., by dropping or adding nodes or edges.

However, all the above-mentioned steps ignore the temporal evolution of both graph structure and communities which is peculiar to the newly proposed SS-DCD task.

Figure 1 shows how to naively use the classical SS-CD pipeline in an SS-DCD task. For the sake of simplicity, we exemplify the CD process across two consecutive time steps only, namely $t-1$ and t. The temporal graph snapshots G^{t-1} and G^t are separately embedded to generate the candidate communities. Next, the candidate selection and refinement processes are executed independently for every time step. This existing pipeline has two main drawbacks:

- The process of generation of the candidate communities disregards the temporal relations among graph snapshots and related communities. Consequently, the next supervised candidate selection step could be biased.
- The community refinement step is unaware of the outcomes of the supervised community detection and refinement steps. Hence, it potentially disregards all past (labeled or detected) community updates as well as the temporal relations with the surrounding network.

Let us consider, for example, the labeled community $\hat{c}^{t-1}$ at time step $t-1$ consisting of nodes v_1 and v_3 ($\hat{c}^{t-1} = \{v_1, v_3\}$). The path connecting v_1 and v_3 changes at time step t by including also node v_4. Knowing both the past community composition and the new network structure allows the CD algorithm to learn the temporal relation with its updated version $\hat{c}^t = \{v_1, v_3, v_4\}$. Analogously, the exclusion of node v_2 from $\hat{c}^{t-1}$ is a valuable hint for the definition of its updated version at time step t.

To overcome the aforesaid limitations, we propose a new SS-DCD pipeline (see Figure 2). Compared to the standard pipeline, we incorporate the following two additional features making the SS-CD pipeline *end-to-end time-aware*.

Time-aware graph embeddings Rather than learning independent embedding matrices for every snapshot t in $\mathcal{G}$, we compute a time-aware embedding representation H^t of the previous graph snapshots $G^1, \ldots, G^t$ capturing time-varying properties of the network graph structure. The embedding representation is incrementally updated at each time step and, importantly, is directly fed to the supervised candidate selection step to allow time-aware semi-supervision (see the *time-aware supervised candidate selection* step in Figure 2).

The designed SS-DCD pipeline is agnostic to the encoder used to compute the embedding matrix. For incremental learning of node embeddings, our current implementation leverages the ROLAND node embeddings [13] (more details are given in Section 5.3).

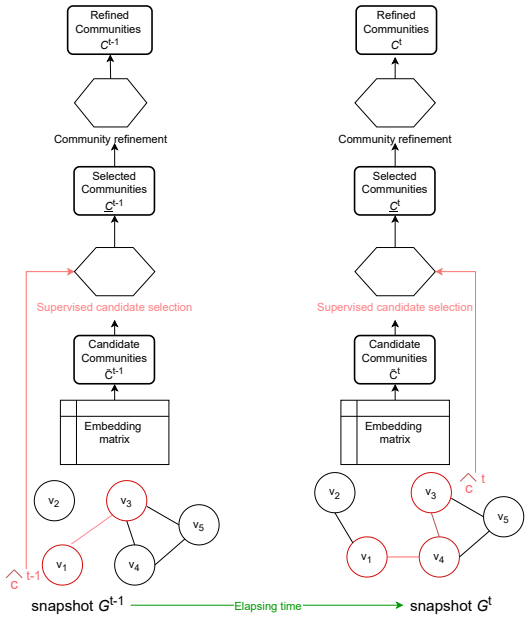

Figure 1: Separate application of the existing SS-CD pipeline to two consecutive time steps.

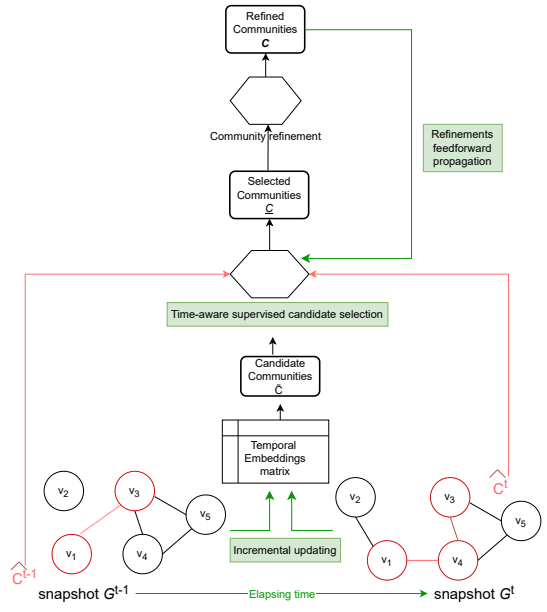

Figure 2: The proposed SS-DCD pipeline applied to two consecutive time steps.

Feed-Forward propagation of community refinements To tailor community refinement to a dynamic scenario (1) We leverage time-aware embeddings to also consider the time-variant properties of the nodes composing the selected community to be refined; (2) We apply a feed-forward propagation of the refined communities' information learnt at time step t across the subsequent time steps.

Given a selected community $\underline{c}^t$, we revise its node composition using a Reinforcement Learning (RL) approach. Let $N^t_{\underline{c}}$ be the subset of G^t nodes belonging to the neighborhood of $\underline{c}^t$. We define the initial state of the RL as the union of the community with its neighborhood ($\underline{c}^t \cup N^t_{\underline{c}}$). The state representation consists of the time-aware embeddings of the composing nodes, i.e., $\mathrm{H}_{\underline{c}^t \cup N^t_{\underline{c}}}$

Similar to CLARE [9], we define two policy networks, consisting of separate MultiLayer Perceptrons, to decide whether to execute any of the following refinement actions on the community $\underline{c}^t$:

- *Expansion*, which adds a new node to the community taken from its neighborhood;
- *Reduction*, which excludes nodes that are already part of the community,

Each action is rewarded according to the F1-Score improvement achieved compared to the previous iteration.

The *ExpMLP* network returns the probabilities of adding to $\underline{c}^t$ a new node from the community neighborhood $N^t_{\underline{c}}$, whereas *ReductMLP* estimates the likelihood of removing any node from $\underline{c}^t$. To make the community refinement process time-aware, both networks are fed with the time-aware embeddings. The action probability distributions are defined as follows.

$$P(\text{action=reduction of } \underline{c}^t) = \text{softmax}\Big(\mathit{ReductMLP}(\mathrm{H}_{\underline{c}^t \cup N^t_{\underline{c}}})\Big)$$

$$P(\text{action=expansion of } \underline{c}^t) = \text{softmax}\Big(\mathit{ExpMLP}(\mathrm{H}_{\underline{c}^t \cup N^t_{\underline{c}}})\Big)$$

The aforesaid probabilities distributions are propagated to the supervised community selection stage to initialize the CD process at the next time step (see the *refinements' feedforward propagation* arrow in Figure 2).

4. Dynamic graphs

In this section, we introduce the generator used to create dynamic graphs with time-evolving labeled communities and the datasets used in the experiments.

4.1. Synthetic Generator of Dynamic Graphs with Labeled Communities

CD algorithms are commonly tested on both real-world and synthetic data [1]. *Real benchmarks* including ground-truth communities (e.g., [17, 18]) are mostly static. Few of them consist of dynamic graphs with annotations (e.g., [19, 20]) but include a relatively low number of ground-truth communities (< 10) and nodes (on average < 100) thus hindering their use for testing Deep Learning techniques. *Synthetic generators* (e.g., [21, 16]) provide ground-truth communities generated by reference external models, thus making the process of semi-supervision unrealistic.

To bridge the gap, we extend a synthetic generator of dynamic graphs [21] to simulate the temporal variations of real graphs and ground-truth communities designed for static scenarios. The cornerstones of our synthetic graph generator, namely *DynamizeGraph*, are enumerated below.

- We consider real graphs and ground-truth communities as initial snapshots (at time step 1).
- We simulate the temporal evolution of the real graph and its ground-truth communities by hiding or showing nodes, edges, or labeled communities.

Table 1
Main dataset statistics: average number of vertices and edges per snapshot, the average number of ground-truth communities per snapshot (train + test), the average modularity per snapshot, and the dynamic graph stability computed according to [16].

Dataset	#Nodes	#Edges	#Commun.	Modularity	Stability
Real initial network and communities - Dynamics injection with high stability					
email-Eu-core	802	10590	40	0.31	0.97
Amazon	13465	31233	1141	0.99	0.98
Youtube	28861	124404	2373	0.22	0.97
Real initial network and communities - Dynamics injection with low stability					
email-Eu-core	550.6	2726.9	38.7	0.65	0.80
Amazon	9580.0	16729.9	1051.6	0.99	0.81
Youtube	16158.5	30836.1	2107.4	0.55	0.74
Purely synthetic					
Syn_Const	5279.3	18894.0	500.0	0.82	0.99
Syn_Growth	3489.6	12594.8	501.0	0.88	0.93
Syn_Shrink	4079.6	19735.0	500.5	0.84	0.98

- We apply graph transformations from one snapshot to another that are related to either specific nodes/edges (*micro-operations*) or entire communities (*macro-operations*).

A Python implementation of *DynamizeGraph* is also available, for research purposes, in the official project repository.

4.2. Datasets

We run our experiments on nine different datasets, six of them are generated by *DynamizeGraph* starting from real graphs and labeled communities whereas the remaining ones are purely synthetic and generated by DANCer [21]. Table 1 summarizes the main dataset statistics. Given the ground truth communities, we consider 50% of them as training labeled communities and the remainder 50% of them as test labeled communities (see Section 5.1).

Real graphs and communities We rely on three real networks with ground-truth communities retrieved from SNAP [17], i.e., *email-Eu-core*, *com-Amazon*, and *com-Youtube*. The real networks are all static but show rather different characteristics. Specifically, email-Eu-core is a denser yet smaller network including roughly 20 nodes per ground-truth community whereas com-Amazon and com-Youtube are sparser yet much larger datasets where communities consist of approximately 10 nodes each. In terms of network modularity, real graphs also significantly differ from each other: com-Amazon has a very high modularity whereas email-Eu-core and com-Youtube show fairly low modularity values. As discussed later on, the lower the modularity the more complex the CD task in the absence of appropriate supervision.

Injection of network dynamics We extend the real static graphs and ground-truth communities under two different dynamic settings, i.e., *low* or *high stability*. In compliance with [16], we define the *stability* as the average difference in Adjusted Mutual Information of the communities [22] between two consecutive time steps. The higher the stability the lower the strength of the dynamics in the network communities across consecutive graph snapshots. We expect to achieve higher benefits from our time-aware approach on dynamic graphs with relatively (but not excessively) low stability. As discussed in Section 5, the results meet the expectation.

Purely synthetic dynamic graphs We generate three dynamic graphs whose snapshots have different sizes, i.e., *Syn_Const* shows a roughly constant number of nodes per snapshot, in *Syn_Growth* the number of nodes per snapshot increases over time, whereas in *Syn_Shrink* shows an opposite trend.

5. Experimental results

All the experiments were conducted on a single NVIDIA Tesla V100 SXM2 GPU with 32 GB memory.

5.1. Performance metrics

We evaluate SS-CD performance using the following three established metrics: *F1 score*, *Jaccard score*, and *Overlapping Normalized Mutual Information* (ONMI, in short). In all cases, we use a set of labelled test communities $\hat{\mathcal{C}}_{test}$, with no intersection with the training ones $\hat{\mathcal{C}}$. To cope with dynamic scenarios all the scores are averaged over all snapshots.

The F1 and Jaccard scores are metrics for comparing test ground-truth and predicted communities in SS-CD [23, 24, 25, 9]. The higher the scores the more accurate the community matching (in whatever direction). To more deeply analyze the impact of graph modularity on CD performance, we also use alternative weighted versions of the *F1* and *Jaccard scores*. Since we are particularly interested in exploring the capabilities of CD methods to leverage the information extracted from labeled communities, we weight the modularity $\mathrm{m}(\hat{c}^t)$ of each test community $\hat{c}^t$ inversely proportional to their modularity, the lower the modularity of a test community, the lower its predictability in the absence of supervised knowledge, the higher its matching contribution to the overall score. Finally, we use the *Overlapping Normalized Mutual Information (OMNI)* [26]. It is a rescaled version of the Mutual Information between the predicted and test sets of communities at time step t.

5.2. Baselines

As baseline methods, we consider the following four state-of-the-art DCD approaches extended to successfully cope with dynamic graphs: DynGEM [12], CTGCN-S [27], CTGCN-C [27], and ROLAND [13]. Specifically, we modify the respective architectures integrating an additional cross-entropy loss term for supervision driven by the labeled

communities. Furthermore, we also consider CLARE [9], which is the latest and best-performing SS-CD approach[1]. The key differences between the extended DCD versions and the SS-CD baseline are that (1) CLARE relies on static order embeddings, and (2) SS-CD also performs community refinement on top of the supervised candidate selection.

5.3. Experimental settings

We test our approach by varying (1) The node embedding strategy (we test Node2Vec [28], ROLAND [13], DynGEM [12], CTGCN-S and CTGCN-C [27]) (2) The policy network architecture in type (we test MLP, GRU, and moving average) and complexity (i.e., we vary the number of attention heads), (3) the dropout rate (between zero and one), and (4) The number of training epochs (up to 2000). Based on a grid search, ROLAND [13] turns out to be the best-performing dynamic graph encoder while a single-head GRU the best policy network. We set the dropout rate to zero, as its impact is negligible, and the number of training epochs to 2000.

For the baseline methods, we adapt the source code released by the papers' authors. All the DCD baselines are trained for 50 epochs, whereas we train CLARE for 30 epochs to perform candidate generation and for 1000 epochs to perform community rewriting. We always use 16-dimension embeddings for DCD models and 64-dimension order embeddings for CLARE.

For all methods, we set the number of expected communities, whenever requested as input parameter, to the number of labeled communities in the training data.

To compute the statistics on the network graphs we use the NetworkX library [29].

5.4. Performance results

Table 2 reports the F1, Jaccard, ONMI Scores achieved by our approach and the baseline methods on the test communities. Among the tested baselines, CLARE performs averagely best on the analyzed datasets and settings confirming the benefits of adopting the complete SS-CD pipeline. Our approach outperforms all the tested baselines, including CLARE, on the YouTube and the synthetic dynamic graphs, it performs best on email-Eu-core in four out of six dataset-metric combinations, whereas ranked second behind CLARE on the Amazon dataset. On average, it shows superior performance on graphs with low modularity (e.g., YouTube) and better suitability for fairly low-stability settings. The reason is that the time-aware approach provides clear benefits when graph snapshots and labeled communities are dynamic (i.e., low stability values) as long as the strength of the dynamics does not invalidate the relevance of the temporal graph relations. For instance, on the same dataset (email-Eu-core) the average F1 Score of our approach soars from 0.1931 to 0.3213 moving from a high-stability setting to a low-stability one. Conversely, on datasets like Amazon where the modularity is high, CD based on the analysis of the network structure is already quite effective. Therefore, the benefits achieved by semi-supervision turn out to be limited.

Figure 3 graphically shows the correlation between average graph modularity and per-dataset F1-score gaps between our approach and CLARE. The result confirms the negative correlation between modularity and usefulness of the time-aware community refinement step. A quantitative comparison in terms of Weighted F1- and Jaccard scores on low-modularity datasets confirms our previous findings (e.g., on YouTube our average Weighted F1-Score is 1.89 vs. 0.9 of CLARE). The achieved results indicate that the time-aware approach turns out to be particularly helpful when the community-level information conveyed by the network graph solely is not enough.

[1]We omit the comparisons with MARS [10] because, to the best of our knowledge, the paper is currently under review and the source code is not available yet.

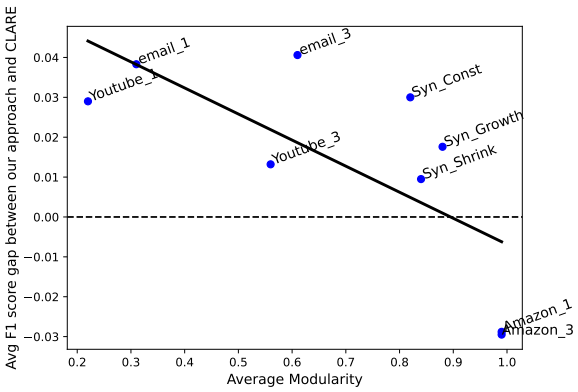

Figure 3: Correlation between average graph modularity and performance gaps between our approach and the SS-CD state-of-the-art model [9]. The higher the modularity the lower the benefits of the time-aware approach.

5.5. Ablation study

We carry out an ablation study to quantify the effect of the choice of the node embedding strategy on dynamic graphs characterized by variable modularity and stability levels. To this end, we compare two alternative state-of-the-art strategies for temporal graph embeddings, i.e., ROLAND [13] and CTGCN [27].

The plots in Figure 4 show the F1-score gaps observed in our approach between the variants with ROLAND and CTGCN graph embeddings. They respectively show the separate influence of graph modularity (plot on the left) and stability (right) on the results of the ablation study. Thanks to the use of hierarchical node states, ROLAND embeddings turn out to be more effective than CTGCN in capturing dynamic graph and community relations. It has shown to be averagely more robust to graphs with low/medium modularity. On datasets with extremely high values of modularity or stability, CTGCN outperforms ROLAND. However, as discussed in Section 5.4, in those extreme cases adopting time-aware approaches to tackle the SS-DCD task is less appealing.

6. Conclusions and Future Work

In this paper, we formulated a new Community Detection task combining Semi-Supervision and dynamic graphs. We extend the existing pipeline for SS-CD by leveraging temporal graph embeddings to capture temporal dynamics in the candidate community generation, selection, and refinement stages. We also propagate the outcomes of two policy networks used in the refinement stage across the subsequent time steps, thus making the supervision aware of the

Table 2
Performance comparison between baselines and our methods on dynamic graphs with different characteristics. For each combination of method, dataset, and performance metric we report the average score over multiple runs and the standard deviations. The best average performance per dataset and score is highlighted in boldface.

	Dataset	DynGEM [12]	CTGCN-C [27]	CTGCN-S [27]	ROLAND [13]	CLARE [9]	Ours
Real initial network and communities - Dynamics injection with high stability							
F1 Score	email-Eu-core	0.3450 ± 0.0821	$\mathbf{0.3815 \pm 0.0210}$	0.1706 ± 0.0167	0.1699 ± 0.0105	0.1591 ± 0.0301	0.1931 ± 0.0263
	Amazon	0.3282 ± 0.0245	0.5488 ± 0.0539	0.1607 ± 0.0183	0.1229 ± 0.0074	$\mathbf{0.6757 \pm 0.0131}$	0.6207 ± 0.0304
	YouTube	0.2134 ± 0.0162	0.1975 ± 0.0217	0.2486 ± 0.0289	0.0867 ± 0.0084	0.4596 ± 0.0122	$\mathbf{0.4665 \pm 0.0229}$
Jaccard Score	email-Eu-core	0.2309 ± 0.0676	$\mathbf{0.2547 \pm 0.0204}$	0.0951 ± 0.0106	0.0944 ± 0.0063	0.0936 ± 0.0199	0.1169 ± 0.0218
	Amazon	0.2209 ± 0.0217	0.4390 ± 0.0566	0.0928 ± 0.0118	0.0667 ± 0.0042	$\mathbf{0.6325 \pm 0.0127}$	0.5699 ± 0.0325
	YouTube	0.1349 ± 0.0125	0.1192 ± 0.0157	0.1612 ± 0.0217	0.0473 ± 0.0050	0.3980 ± 0.0113	$\mathbf{0.4042 \pm 0.0216}$
ONMI	email-Eu-core	0.1173 ± 0.0580	0.0956 ± 0.0285	0.0042 ± 0.0055	0.0000 ± 0.0000	0.0102 ± 0.0165	$\mathbf{0.0254 \pm 0.0297}$
	Amazon	0.0833 ± 0.0258	0.3440 ± 0.0730	0.0084 ± 0.0033	0.0002 ± 0.0004	$\mathbf{0.6164 \pm 0.0114}$	0.5487 ± 0.0352
	YouTube	0.0330 ± 0.0093	0.0177 ± 0.0081	0.0547 ± 0.0159	0.0009 ± 0.0010	0.3596 ± 0.0104	$\mathbf{0.3678 \pm 0.0222}$
Real initial network and communities - Dynamics injection with low stability							
F1 Score	email-Eu-core	0.3032 ± 0.0555	$\mathbf{0.4637 \pm 0.0791}$	0.1979 ± 0.0242	0.2042 ± 0.0137	0.3117 ± 0.0520	$\mathbf{0.3213 \pm 0.0504}$
	Amazon	0.4765 ± 0.0333	0.4657 ± 0.0325	0.1873 ± 0.0320	0.1138 ± 0.0075	$\mathbf{0.6414 \pm 0.0127}$	0.6120 ± 0.0266
	YouTube	0.1426 ± 0.0125	0.2532 ± 0.0409	0.2352 ± 0.0497	0.1041 ± 0.0203	0.4867 ± 0.0244	$\mathbf{0.4912 \pm 0.0307}$
Jaccard Score	email-Eu-core	0.1919 ± 0.0447	$\mathbf{0.3432 \pm 0.0802}$	0.1124 ± 0.0166	0.1176 ± 0.0098	0.2183 ± 0.0495	$\mathbf{0.2206 \pm 0.0509}$
	Amazon	0.3799 ± 0.0399	0.3580 ± 0.0316	0.1100 ± 0.0205	0.0617 ± 0.0042	$\mathbf{0.5935 \pm 0.0147}$	0.5560 ± 0.0324
	YouTube	0.0864 ± 0.0093	0.1596 ± 0.0310	0.1514 ± 0.0349	0.0583 ± 0.0118	0.4252 ± 0.0245	$\mathbf{0.4308 \pm 0.0301}$
ONMI	email-Eu-core	0.0821 ± 0.0593	$\mathbf{0.2188 \pm 0.1012}$	0.0036 ± 0.0075	0.0065 ± 0.0103	0.1447 ± 0.0728	0.1200 ± 0.0792
	Amazon	0.3082 ± 0.0592	0.2640 ± 0.0386	0.0088 ± 0.0045	0.0008 ± 0.0010	$\mathbf{0.5862 \pm 0.0169}$	0.5495 ± 0.0307
	YouTube	0.0147 ± 0.0066	0.0382 ± 0.0198	0.0478 ± 0.0199	0.0022 ± 0.0013	0.3944 ± 0.0312	$\mathbf{0.4013 \pm 0.0367}$
Purely synthetic							
F1 Score	Syn_Const	0.3056 ± 0.0132	0.4778 ± 0.0699	0.1682 ± 0.0084	0.1360 ± 0.0030	0.5947 ± 0.0295	$\mathbf{0.6071 \pm 0.0228}$
	Syn_Growth	0.4074 ± 0.0308	0.5908 ± 0.0869	0.2518 ± 0.0182	0.1856 ± 0.0126	0.7048 ± 0.0227	$\mathbf{0.7134 \pm 0.0183}$
	Syn_Shrink	0.4199 ± 0.0498	0.6443 ± 0.0989	0.2227 ± 0.0245	0.1730 ± 0.0144	0.6860 ± 0.0443	$\mathbf{0.6886 \pm 0.0355}$
Jaccard Score	Syn_Const	0.1899 ± 0.0104	0.3480 ± 0.0641	0.0936 ± 0.0053	0.0735 ± 0.0018	0.4886 ± 0.0314	$\mathbf{0.5064 \pm 0.0271}$
	Syn_Growth	0.2768 ± 0.0294	0.4699 ± 0.0920	0.1489 ± 0.0130	0.1037 ± 0.0078	0.6264 ± 0.0279	$\mathbf{0.6420 \pm 0.0228}$
	Syn_Shrink	0.2905 ± 0.0471	0.5382 ± 0.1138	0.1306 ± 0.0166	$0.0963 \pm 0,0090$	0.5973 ± 0.0547	$\mathbf{0.6032 \pm 0.0425}$
ONMI	Syn_Const	0.0258 ± 0.0094	0.2114 ± 0.0740	0.0008 ± 0.0013	0.0000 ± 0.0000	0.4708 ± 0.0437	$\mathbf{0.4944 \pm 0.0362}$
	Syn_Growth	0.1150 ± 0.0402	0.3763 ± 0.1277	0.0100 ± 0.0048	0.0000 ± 0.0000	0.6337 ± 0.0336	$\mathbf{0.6493 \pm 0.0247}$
	Syn_Shrink	0.1368 ± 0.0615	0.4555 ± 0.1473	0.0090 ± 0.0053	0.0007 ± 0.0011	0.5965 ± 0.0614	$\mathbf{0.6040 \pm 0.0453}$

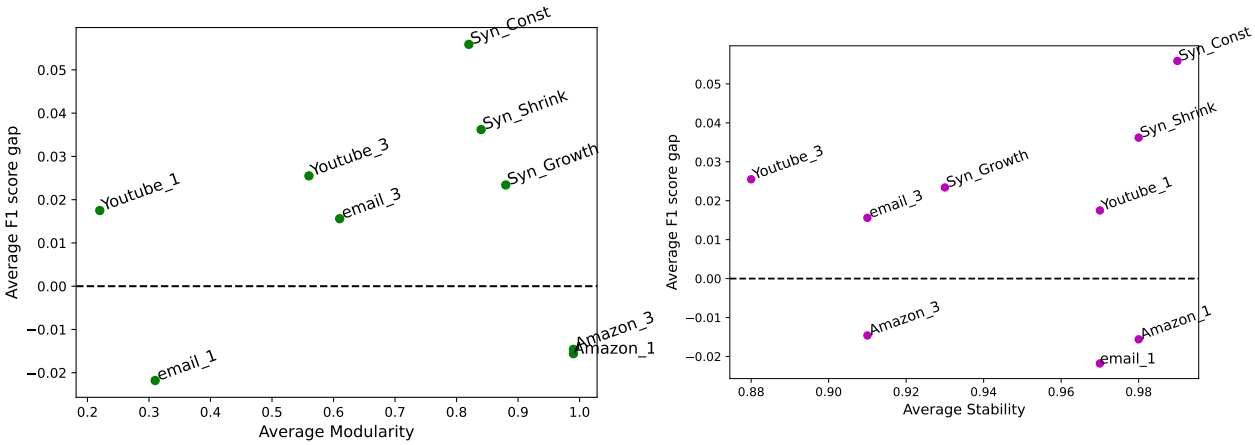

Figure 4: Gap between the F1-Scores of our approach using ROLAND and CTGCN embeddings. Effects of average graph modularity (left hand-side) and average graph stability (right).

temporal relations of communities with the past, partially labeled, graph snapshots. The main takeaways from the empirical analysis are: (i) the effectiveness of the proposed pipeline on low-modularity graphs and low-stability graphs, (ii) the importance of the community refinement stage as in the classical SS-CD task, and (iii) the little influence of the number of nodes on computational time of the community refinement stage, suggesting to optimize the cost of graph embedding to efficiently adopt time-aware strategies in large networks.

Our future research agenda will encompass: (i) the extension to network graphs with node attributes, which might also change over time, (ii) the study of scalable approaches to SS-DCD in the steps of graph embedding, candidate community generation, but also the community refinement stage, and (iii), and the adoption of Contrastive Learning architectures to generate input embeddings (replacing Graph Neural Networks), thus limiting the complexity and need for large sets of annotations.

Acknowledgments

This work has been partially supported by the Spoke 1 "FutureHPC & BigData" of ICSC - Centro Nazionale di Ricerca in High-Performance-Computing, Big Data and Quantum Computing, funded by European Union - NextGenerationEU.

References

[1] X. Su, S. Xue, F. Liu, J. Wu, J. Yang, C. Zhou, W. Hu, C. Paris, S. Nepal, D. Jin, Q. Z. Sheng, P. S. Yu, A comprehensive survey on community detection with deep learning, IEEE Transactions on Neural Networks and Learning Systems 35 (2024) 4682–4702. URL: http://dx.doi.org/10.1109/TNNLS.2021.3137396. doi:10.1109/tnnls.2021.3137396.

[2] P. Chunaev, Community detection in node-attributed social networks: a survey, Computer Science Review 37 (2020) 100286. URL: https://www.sciencedirect.com/science/article/pii/S1574013720303865. doi:https://doi.org/10.1016/j.cosrev.2020.100286.

[3] X. Huang, D. Chen, T. Ren, D. Wang, A survey of community detection methods in multilayer networks, Data Min. Knowl. Discov. 35 (2021) 1–45. URL: https://doi.org/10.1007/s10618-020-00716-6. doi:10.1007/s10618-020-00716-6.

[4] M. Rostami, M. Oussalah, K. Berahmand, V. Farrahi, Community detection algorithms in healthcare applications: A systematic review, IEEE Access 11 (2023) 30247–30272. doi:10.1109/ACCESS.2023.3260652.

[5] M. Girvan, M. E. J. Newman, Community structure in social and biological networks, Proceedings of the National Academy of Sciences 99 (2002) 7821–7826. doi:10.1073/pnas.122653799.

[6] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, Journal of Statistical Mechanics: Theory and Experiment 2008 (2008) P10008. URL: http://dx.doi.org/10.1088/1742-5468/2008/10/P10008. doi:10.1088/1742-5468/2008/10/p10008.

[7] N. Mehta, L. Carin, P. Rai, Stochastic blockmodels meet graph neural networks, in: K. Chaudhuri, R. Salakhutdinov (Eds.), Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, PMLR, 2019, pp. 4466–4474.

[8] J. Yang, J. Leskovec, Overlapping communities explain core–periphery organization of networks, Proceedings of the IEEE 102 (2014) 1892–1902. doi:10.1109/JPROC.2014.2364018.

[9] X. Wu, Y. Xiong, Y. Zhang, Y. Jiao, C. Shan, Y. Sun, Y. Zhu, P. S. Yu, Clare: A semi-supervised community detection algorithm, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 2059–2069. URL: https://doi.org/10.1145/3534678.3539370. doi:10.1145/3534678.3539370.

[10] L. Haonan, L. Xiaoyu, H. Linmei, J. Li, S. Xian, Z. Linhao, W. Kaiwen, W. Hongqi, Mars: An iterative matching and rewriting model for semi-supervised community detection, Available at SSRN 4757429 (2024).

[11] K. Berahmand, Y. Li, Y. Xu, A deep semi-supervised community detection based on point-wise mutual information, IEEE Transactions on Computational Social Systems 11 (2024) 3444–3456. doi:10.1109/TCSS.2023.3327810.

[12] P. Goyal, N. Kamra, X. He, Y. Liu, Dyngem: Deep embedding method for dynamic graphs, CoRR abs/1805.11273 (2018). URL: http://arxiv.org/abs/1805.11273. arXiv:1805.11273.

[13] J. You, T. Du, J. Leskovec, Roland: Graph learning framework for dynamic graphs, 2022. arXiv:2208.07239.

[14] Z. Wang, C. Wang, C. Gao, X. Li, X. Li, An evolutionary autoencoder for dynamic community detection, Science China Information Sciences 63 (2020) 212205. URL: https://doi.org/10.1007/s11432-020-2827-9. doi:10.1007/s11432-020-2827-9.

[15] L. Fan, S. Xu, D. Liu, Y. Ru, Semi-supervised community detection based on distance dynamics, IEEE Access 6 (2018) 37261–37271. doi:10.1109/ACCESS.2018.2838568.

[16] G. Paoletti, L. Gioacchini, M. Mellia, L. Vassio, J. M. Almeida, Benchmarking evolutionary community detection algorithms in dynamic networks, in: 4th Workshop on Graphs and more Complex structures for Learning and Reasoning (GCLR) at AAAI 2024, Cornell Tech, 2024, p. 1–8. URL: https://arxiv.org/abs/2312.13784.

[17] J. Leskovec, A. Krevl, SNAP Datasets: Stanford large network dataset collection, http://snap.stanford.edu/data, 2014.

[18] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, Physical Review E 78 (2008). URL: http://dx.doi.org/10.1103/PhysRevE.78.046110. doi:10.1103/physreve.78.046110.

[19] P. Vanhems, A. Barrat, C. Cattuto, J.-F. Pinton, N. Khanafer, C. Régis, B.-a. Kim, B. Comte, N. Voirin, Estimating potential infection transmission routes in hospital wards using wearable proximity sensors, PLOS ONE 8 (2013) 1–9. URL: https://doi.org/10.1371/journal.pone.0073970. doi:10.1371/journal.pone.0073970.

[20] R. Mastrandrea, J. Fournet, A. Barrat, Contact patterns in a high school: A comparison between data collected using wearable sensors, contact diaries and friendship surveys, PLOS ONE 10 (2015) 1–26. URL: https://doi.org/10.1371/journal.pone.0136497. doi:10.1371/journal.pone.0136497.

[21] C. Largeron, P.-N. Mougel, O. Benyahia, O. R. Zaïane, Dancer: dynamic attributed networks with community structure generation, Knowledge and Information Systems 53 (2017) 109–151.

[22] X. V. Nguyen, J. Epps, J. Bailey, Information theoretic measures for clusterings comparison: is a correction for chance necessary?, in: Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009, volume 382, ACM, 2009, pp. 1073–1080. doi:10.1145/1553374.1553511.

[23] J. Yang, J. J. McAuley, J. Leskovec, Community detection in networks with node attributes, in: H. Xiong, G. Karypis, B. Thuraisingham, D. J. Cook, X. Wu (Eds.), 2013 IEEE 13th International Conference on Data Mining, Dallas, TX, USA, December 7-10, 2013, IEEE Computer Society, 2013, pp. 1151–1156. URL: https://doi.org/10.1109/ICDM.2013.167. doi:10.1109/ICDM.2013.167.

[24] T. Chakraborty, A. Dalmia, A. Mukherjee, N. Ganguly, Metrics for community analysis: A survey, 2016. arXiv:1604.03512.

[25] Y. Jia, Q. Zhang, W. Zhang, X. Wang, Communitygan: Community detection with generative adversarial nets, 2019. arXiv:1901.06631.

[26] A. F. McDaid, D. Greene, N. Hurley, Normalized mutual information to evaluate overlapping community finding algorithms, 2013. arXiv:1110.2515.

[27] J. Liu, C. Xu, C. Yin, W. Wu, Y. Song, K-core based temporal graph convolutional network for dynamic graphs, CoRR abs/2003.09902 (2020). URL: https://arxiv.org/abs/2003.09902. arXiv:2003.09902.

[28] J. Leskovec, L. Backstrom, R. Kumar, A. Tomkins, Microscopic evolution of social networks, in: KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, New York, NY, USA, 2008, pp. 462–470. URL: http://dx.doi.org/10.1145/1401890.1401948. doi:10.1145/1401890.1401948.

[29] A. A. Hagberg, D. A. Schult, P. J. Swart, Exploring network structure, dynamics, and function using networkx, in: G. Varoquaux, T. Vaught, J. Millman (Eds.), Proceedings of the 7th Python in Science Conference, Pasadena, CA USA, 2008, pp. 11 – 15. URL: http://conference.scipy.org/proceedings/SciPy2008/paper_2/.