Dynamic Local Community Detection with Anchors

Konstantinos Christopoulos[1], Georgia Baltsou[2], and Konstantinos Tsichlas[1]

[1] Department of Computer Engineering and Informatics, University of Patras, Greece,
kchristopou@upnet.gr,ktsichlas@ceid.upatras.gr
[2] School of Informatics, Aristotle University of Thessaloniki, Greece
georgipm@auth.gr

Abstract. Community detection is a challenging research problem, especially in dynamic networks, since in this case communities cannot remain stable as they evolve. In evolving networks new communities may emerge and existing communities may disappear, grow or shrink. There are many cases where someone is more interested in the evolution of a particular community, to which an important node belongs, rather than in the global partitioning of a dynamic network. However, due to the drifting problem where one community can evolve into a completely different one, it is difficult to track the evolution of communities. Our aim is to identify the community that contains a node of particular importance, called anchor, and its evolution over time. The framework we propose circumvents the identity problem by allowing the anchor to define the core of the relevant community partially or fully. Preliminary experiments with synthetic datasets demonstrate the positive aspects of the proposed framework in identifying communities with high accuracy.

Keywords: local community detection;networks;dynamic;anchor

1 Introduction

Networks are used to represent entities and their relations for systems of almost any domain like biology, society, transportation etc. In such systems, there is a huge amount of data that is constantly generated. Community detection constitutes an important task of network analysis. It's aim is to uncover groups of densely connected entities called communities. Most of the existing literature is concerned with the global community detection problem, i.e. a whole network's division into communities. However, in many cases we are interested only on the communities around a few particular nodes. For this reason, local community detection has lately attracted scientists' interest. Consequently, there are different cases where global approach is preferred over the local and vice versa. Generally, local community detection is more suitable for discovering the communities for nodes of interest on complex networks with low computational cost. Most existing work on community detection problems is on static networks. Static networks do not change their structure (nodes and edges) over time. However, most real-world networks change rapidly, and sometimes relationships are established only instantaneously. Networks that are time evolving are called dynamic or temporal. Moreover, with the rapid growth of the internet and its applications, real-world network datasets are extremely large, making it unreasonable to process them in their entirety as it must be done in the case of global community detection. One way to model such rapidly changing networks is by assuming that updates/actions[3] come in a streaming fashion. This means that in each time

[3] We use the terms updates and actions interchangeably to refer to a small change in the network, such as edge insertion or/and deletion.

instance one update is performed on the network. In this sense, time is defined with respect to these updates and the life span of an edge is defined by the updates between its insertion and its removal from the graph [9, 2] - that is, we use transaction time as our main notion of time.

In the present work, we focus on the detection of local communities of particular nodes in temporally evolving networks by revising the theoretical framework of [1] and providing preliminary experimental results that verify its effectiveness. More specifically, our goal is to uncover the community evolution of a node of particular interest, called *anchor*. The importance of this node for the evolving community is considered external knowledge, that is, knowledge that cannot be inferred from the structure of the network. This node defines the evolving community and functions as an anchor for the community circumventing in this way the identity problem. As an example, one can think of a football team community in a social network. This community evolves since new fans may connect or existing fans may stop supporting the team. However, the core fans (e.g., ultras) of the team are more stable and in some sense behave as an anchor for this community.

1.1 Contributions

Our present work focuses on the identification of the community of a specific node called anchor, which is assumed to be of particular importance to this community based on external knowledge in temporal networks. To achieve this, we propose a multi-step framework that firstly applies a static algorithm to discover the initial anchor's community and then for each update "near" the anchor its community is updated. We experimentally show the promise of the suggested framework when compared to other methods in synthetic datasets. Our contribution is twofold:

- From a **modeling perspective**, our contribution lies in the introduction of the notion of the anchor node in the local community detection problem in time evolving networks.
- From an **algorithmic perspective**, a general multi-step framework is suggested to be used in order to uncover stable communities of an important node in time evolving networks.

The remaining sections are organized as follows. In Section 2, we review the literature on local community detection in dynamic networks. The proposed framework is described in Section 3. In Section 4, we present experimental results illustrating our algorithmic framework. Finally, we discuss future expansions of the suggested framework and conclude in Section 5.

2 Related Work

Local community detection, which is also known as the seed set expansion problem, has attracted the attention of researchers as it is very common to process only a small part of the network, either because of its large size or because it is dynamic or someone might be interested in focusing on a specific part of the network. Consequently, there have been many different approaches proposed. However, the literature in dynamic networks is much smaller when compared to the case of static networks. In the following, we discuss local community detection algorithms that are closely related to our work, i.e. in dynamic networks processed in a streaming fashion

In [14] the authors adopt the static L-metric approach [3] in order to find dynamic communities in an incremental way. L-metric is a measure based on the assumption that a community has fewer connections to nodes outside of this community. At each snapshot communities are uncovered using information from previous snapshots and at the end communities found in different snapshots are

matched based on their similarity (L-metric). Experiments showed that the method resulted in meaningful communities. In addition, a dynamic seed set expansion method is proposed in [16, 17] where the authors suggest updating the fitness score of each snapshot incrementally. In order to keep a community centered around the seed, their method ensures that the order of fitness scores remains monotonically increasing by tracking the order of nodes added. Experiments showed that the suggested method is quite fast and the performance is better when low-latency updates are required. Furthermore, in [4] a method called PHASR to find the temporal community with the lowest conductance is proposed. This work aims to find communities with stable membership over time. Experiments showed that the suggested method has low runtime and achieves to find high quality communities. Moreover, the authors of [7] use a metric called local fitness to firstly find the starting nodes of a community and run a static algorithm to define the communities in the first snapshot. In the following snapshots, they use a node contribution metric to incrementally reveal communities. Their experiments showed that the proposed method uncovered communities with high accuracy. Finally, to the best of our knowledge, two methods, [10] and [15], have been proposed for local community detection in graph streams. In [10], the CoEuS algorithm is suggested. The constraint of this method lies in the fact that only a single access to the stream is possible and the working memory is limited. Experiments on networks showed that the algorithm is able to discover local communities with high accuracy. More recently, the algorithm called SCDAC that is suggested in [15], seeks an optimal community on the subgraph intercepted by the streaming model. Experiments showed that SCDAC is more effective and efficient than CoEuS in real networks.

3 Dynamic Local Community Detection with Anchors

3.1 Preliminaries and Problem Formulation

Let $G = (V, E_t)$ be a dynamic network which is composed of a node set V and a set of time-stamped edges E_t. E_t represents interactions among the nodes at time t, where $t \in \mathbb{N}$, generated by an interaction streaming source S. The interaction streaming source S may produce new interactions between nodes which can be either already part of the network or new ones. In particular, it forms a sequence of actions in which interactions flow in streams over time. In this paper, we assume that an action may be an edge insertion or an edge deletion. As a result, the corresponding network's communities also change as the network evolves. Dynamic community detection is the process by which we can observe the evolution of the network's communities.

Given a node A called anchor, the network G and an interaction generator S, our aim is to discover the community C which includes A. We assume that the anchor is of particular importance for the community (external knowledge) it belongs and thus, it operates as a reference point for this community, i.e., anchor defines in a sense the community it belongs to.

In order to minimize the avalanche effect[4] [13], we suggest to limit the community updating only to an influence range around the anchor. The *influence range*, R, defines the radius of the ball centered around the anchor in each time instance of the evolving network. In this ball, all nodes with maximum length of their shortest path to the anchor $\leq R$ are contained. For example, an influence range equal to 1 means that we should update the community structure considering both the anchor as well as its adjacent nodes. Generally, a high influence range value would increase the process demands, since a larger network area would be examined.

[4] The avalanche effect corresponds to the phenomenon where communities can experience substantial drifts compared to what a static algorithm computes in each time instance.

Besides, with a view to discover the most stable anchor's community, we can use a node rewarding method. That is, for each update, we suggest to reward the edges in the anchor's influence range by a weight increase. In our setting, we use three different rewarding methods. Assuming that R is the influence range and d the distance between the anchor and a node, we define the rewarding methods as follows:

1. *Dynamic reward1*: $w = 2R - 2(d-1)$. For instance, if $R = 3$, the edges of the anchor to its adjacent nodes get a weight of $w = 6$, as $d = 1$. Consequently, all the edges of the anchor's adjacent nodes to their adjacent nodes, where the distance from anchor is $d = 2$, get a weight of $w = 4$, and so on until $d = R$.
2. *Dynamic reward2*: $w = R^{R/d}$. Similar to the previews one, if $R = 3$, the edges of the anchor to its adjacent nodes get a weight of $w = 27$, as $d = 1$. Consequently, all the edges of the anchor's adjacent nodes to their adjacent nodes, where the distance from anchor is $d = 2$, get a weight of $w = \sqrt{27}$, and so on until $d = R$.
3. *Dynamic reward3*: this is a rewarding system that takes into account in a very simple manner the history of an edge. When a new edge arrives that its nodes have minimum distance d from the anchor, we initially set its weight to $w = R - d + 1$. Then, if the edge persists after a batch of y actions we assign an extra unit reward to it. y is a user-determined parameter. If an edge is reinserted and lies in the influence range of the anchor then the reward that receives is estimated by the ratio of edge appearances to total number of actions (edge insertion or deletion) in the influence range.

The quality of a community C can be measured by different quality metrics. We use three such metrics in order to objectively evaluate the performance of our suggested framework. The first quality metric is f_{monc}, which is defined as the ratio of the sum of the degrees of internal nodes to other nodes within the community divided by the total sum of the degrees of nodes in C [8]:

$$f(C)_{monc} = \frac{2k_{in}^{C} + 1}{(2k_{in}^{C} + k_{out}^{C})^{\alpha}},$$

where k_{in}^{C} and k_{out}^{C} are the total internal and external degrees of the nodes of community C, and α is a positive real-valued parameter, controlling the size of the communities. The second quality metric we choose to use is LWP which is the ratio of interior edges to edges leaving community C defined in [11] as:

$$LWP(C) = \frac{k_{in}^{C}}{k_{out}^{C}}.$$

The third community quality metric that we use is *conductance* as defined in [6]. For community C and its complement $\overline{C} = V \setminus C$ conductance is defined as:

$$cond(C) = \frac{c(C)}{min(l(C,V), (\overline{C},V))},$$

where $c(C)$ is equal to $cut(C)$, which is defined as the number of edges between nodes in C and nodes in its complement $\overline{C}$. $l(A,B)$ is the number of edges between nodes in A and nodes in B

3.2 Proposed Framework

We assume network G with node set V and edge set E where edges are unweighted. Our method is divided into five steps. The first two are the initialization and the rest the streaming process.

Initialization: The first step of the suggested framework, is to apply weights on the edges according to the anchors influence range. More precisely, we apply a weighting scheme that rewards the edges being closer to the anchor. The depth till which edges are rewarded starting from the anchor is predefined by the chosen influence range R.

The second step of our proposed framework is the application of a greedy static algorithm [8] on the initial state of network G, at timestamp defined as $t = 0$. At this timestamp, the community C contains only the anchor (u_0), and then new nodes are iteratively added. A node is added to C (e.g. u_1) only if the fitness score (e.g. $f(C)_{monc}$) is increased ($f_0 < f_1$). The static algorithm terminates when the fitness score can not be increased anymore. At the end of the second step, a community evolution sequence[5] is created and interior($K_{n,in}$)/external($K_{n,out}$) community edges are also recorded.(i.e. the sequence in which each node entered the community, see Table 1).

Table 1: Community evolution depending on the fitness scores order.

Sequence	0	1	2	...	n
Nodes	u_0	u_1	u_2	...	u_n
Interior edges	$K_{0,in}$	$K_{1,in}$	$K_{2,in}$	...	$K_{n,in}$
External edges	$K_{0,out}$	$K_{1,out}$	$K_{2,out}$	...	$K_{n,out}$
Fitness score	f_0	f_1	f_2	...	f_n

Streaming process: In the third step of the process, a stream of network updates i is applied. These updates can be either edge insertions or deletions. If i occurs in the anchors' influence range: 1) influence range has to be recomputed and 2) edge weights have to be updated considering a rewarding method. Consequently, if i occurs in the anchor's community C or the updated weights affects it, then the C quality measure has to be updated. More precisely, the measure has to be recalculated and if after this recalculation the fitness scores are not anymore in an increasing order, then the node that disrupts this order must be removed and interior/border edges must be modified as well. Then, the same procedure is applied for community nodes that are neighbors of the removed node, and repeat as long as there are neighbors that are not affected by these changes. Even if i does not occur in the anchors' influence range, we should still check the anchors' community because is may extended beyond the influence range. In the fourth step, after fixed-size batches of actions: 1) we check if the sequence of fitness scores is in increasing order and if not, we remove all nodes from the sequence from the leftmost violation up to end and 2) we apply the static algorithm on the updated community of the anchor. The number of actions in each batch is calculated as the ratio of the total number of actions to a user-defined constant value x. That is, the static algorithm will run no more than x times. We need to note here that the more times we choose to run the static algorithm, the more accurate is the outcome of the process. However, the computational cost is higher. Thus, after experimental evaluation, we conclude that the value of the constant should be equal to 20. Figure 1 depicts the steps of the proposed framework.

[5] If the quality metric is Conductance then the fitness score must be decreased and so, the fitness scores should be in a decreasing order.

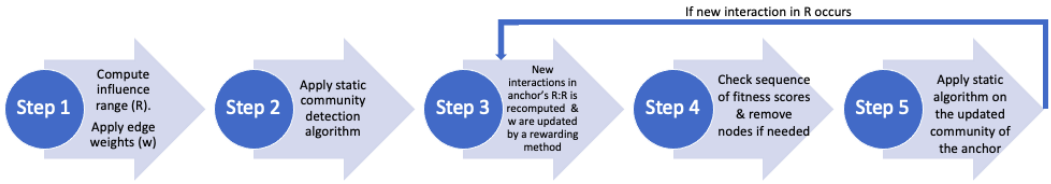

Fig. 1: The proposed approach for dynamic local community detection with anchors.

4 Experiment Design

4.1 Datasets

The synthetic datasets we use in our experiments are generated by RDyn [12], an approach capable of generating dynamic networks that respect well-known real-world network properties along with time-dependent ground truth communities with adjustable quality, i.e., allowing both merging and splitting communities. The generator contains two significant, user-defined, parameters. The first is the number of nodes of the produced dynamic network and the second is the number of iterations. Each iteration consist of a batch of actions (edge insertion/deletion) and the number of these actions are not necessary equal in every iteration. The first iteration of each synthetic dataset is utilized for the purpose of the creation of the initial graph. So that, in each dataset the actions of first iteration are not taken into consideration to the total amount of actions. In our experiments, we use three different datasets produced by the RDyn generator. The basic characteristics of these datasets are described in Table 2.

Table 2: Synthetic datasets with number of nodes, iterations, initial/final edges and actions.

Dataset	Nodes	Iterations	Initial edges	Final edges	Actions
$SD1$	100	100	99	478	6268
$SD2$	500	1000	495	1648	40939
$SD3$	5000	1000	4917	25590	246191

4.2 Evaluation Metrics

To evaluate our proposed framework, we compare the results of our community detection with the ground truth communities produced by the synthetic dataset generator. However, an eligible (to some extent) argument against using the ground truth communities of the synthetic generator is that the discovered community is affected by the anchor. To this end, on the one hand we tried to setup the generator so that communities are not so intertwined while on the other hand we are more interested in comparing the methods between each other rather than looking at values of the metrics w.r.t. the ground truth. The evaluation metrics that are suitable for our purposes are precision, recall, and the F1 score. Precision is the ratio of elements found correctly to the total number of

elements found. Recall is the proportion of relevant elements that were successfully retrieved. The F1 score is the harmonic mean of precision and recall [5]. The harmonic mean is used instead of the simple average because in this way the extreme values are penalised.

4.3 Experiment Results

In our experiments we use three different quality metrics, f_{monc}, *Conductance* and LWP. The node we use as an anchor for each experiment is determined based on its degree centrality. That is, in the first dataset we use a node with low degree as an anchor, in the second we choose two nodes with medium degree, and in the third we use a node with high degree. In the experiments with the first dataset, we use the three quality metrics, while for the others we use the f_{monc} since it provides the best results. We choose the user-defined parameter for f_{monc} to be $a = 1$ and the influence range equal to 2.

Regarding the first synthetic graph, Figures 2, 3 and 4 shows the results of the three quality metrics. The values on the x-axis represents the number of actions. The graph generator provides the graph partition after one or more iterations, where in each iteration the number of actions are not equal. As a consequence, each interval on the x-axis consist of the same number of iterations but different number of actions. The vertical lines on the x-axis shows the events (merge or split) that occurred in the ground truth communities, right after an action. One of these communities contains our anchor.

Analyzing the experimental results, we find that all three dynamic methods with rewards (Dynamic Reward1, Dynamic Reward2, and Dynamic Reward3) outperform dynamic method without rewards [17] (Dynamic1). More precisely, the critical point of community evolution is the time when the first event (merge) occurs. Before this point, recall and precision were very low. There is an explosion in the recall metric and as a result, an improvement in the F1 score. In addition, between actions 4969 and 5786, the static algorithm is activated and as a consequence, the nodes affiliation in anchor's community is nearly the same with those in ground truth community. This is true for all quality metrics and for all dynamic methods except for Dynamic1. For the latter method, we observe large variations in recall values, which is reflected in the F1 score. In more detail, looking carefully in dataset we can observe the following: 1) just before the action 5700, few edges (not related directly to anchor) that belong to the influence range of our community are inserted and at the same time the static algorithm is activated. As a consequence, our methods using conductance as a quality metric, take advantage of rewards and remain stable with high recall values. On the contrary, the same is not true for Dynamic1. 2) In Figure 3 using LWP as a quality metric, we notice the same outcome. Here, our reference point is the action 5896. Again, before this action we observe few edge insertions in the influence range with similar results as before. 3) On the other hand, for both quality metrics the recall values of Dynamic1 method rises between actions 5928 and 6091. Here, we notice an edge insertion between our anchor an its adjacent node. This action helps the method without rewards to reach the high recall values of other methods. Lastly, f_{monc} provides more stable and better results but still our methods outperform Dynamic1.

Next, we consider the second synthetic graph containing 500 nodes. Figures 5 and 6 shows the results of the three evaluation metrics using the f_{monc}. Here we run the experiment twice with two

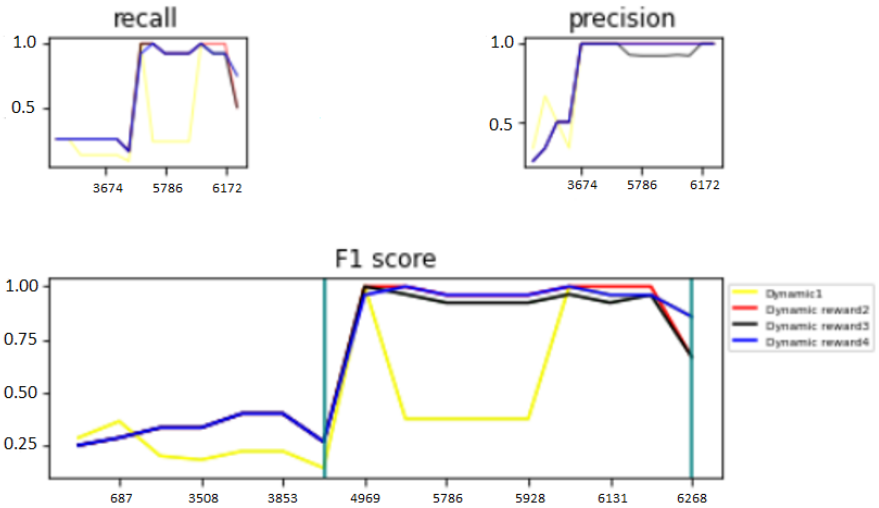

Fig. 2: Results using the Synthetic dataset with 100 nodes and Conductance as a quality metric.

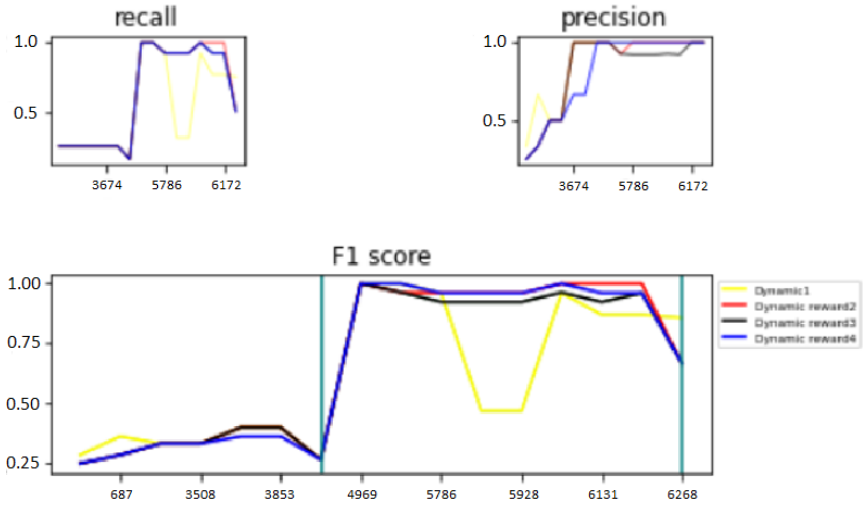

Fig. 3: Results using the Synthetic dataset with 100 nodes and LWP as a quality metric.

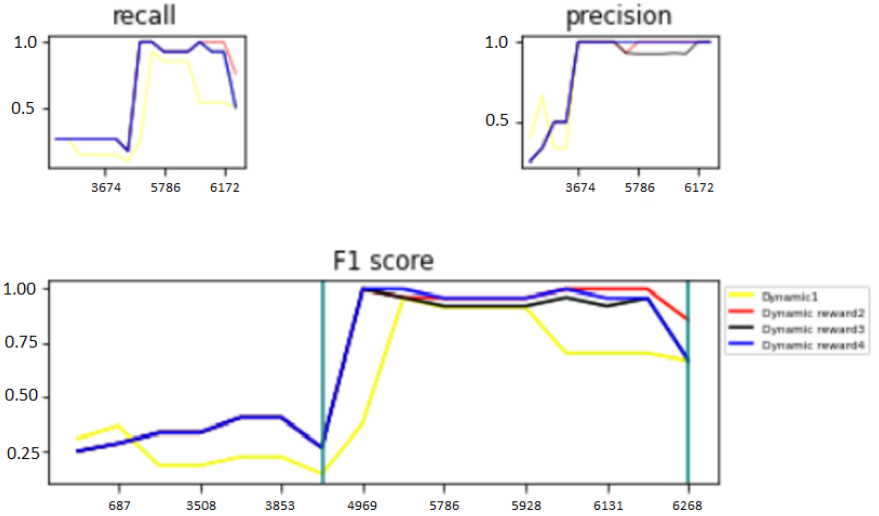

Fig. 4: Results using the Synthetic dataset with 100 nodes and f_{monc} as a quality metric

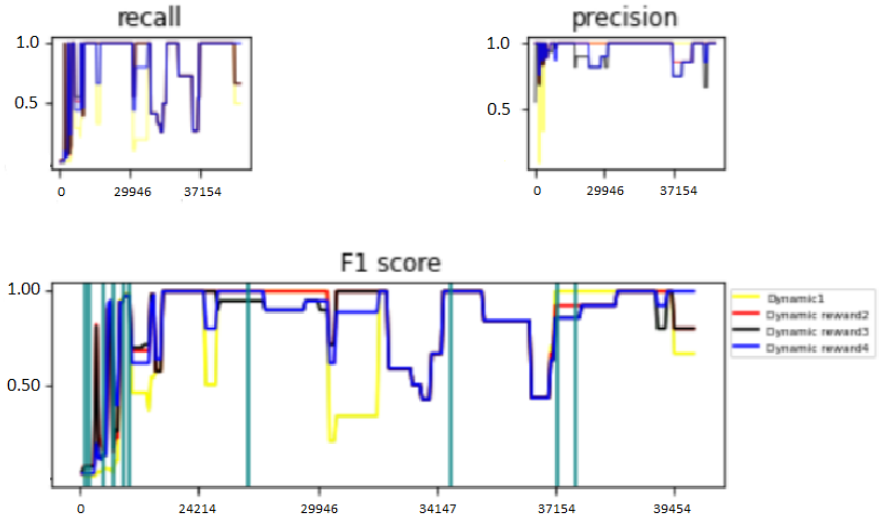

Fig. 5: Results using the Synthetic dataset with 500 nodes and f_{monc} as a quality metric.

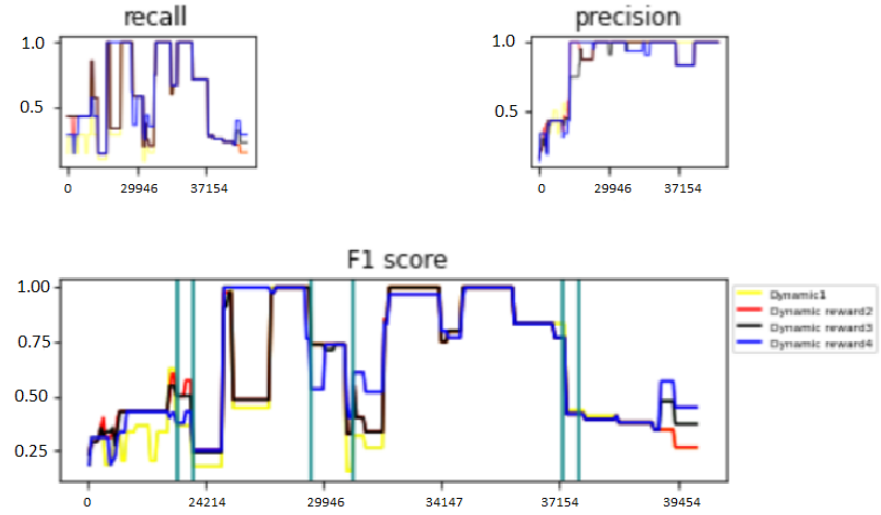

Fig. 6: Results using the Synthetic dataset with 500 nodes and f_{monc} as a quality metric

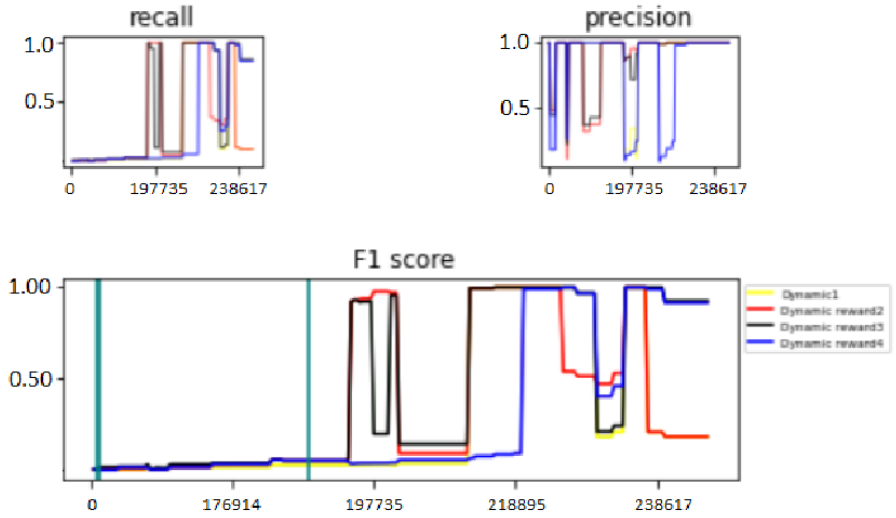

Fig. 7: Results using the Synthetic dataset with 5000 nodes and f_{monc} as a quality metric

different anchors. In Figure 5 we see 10 events. In the first actions, 6 events take place, which affect the recall and precision of each method. More specifically, all dynamic methods have extreme ups and downs, and Dynamic1 being the most affected. After these events, it is clear to see that our methods outperform Dynamic1. The small time interval in which Dynamic1 outperform the other methods is due to an edge deletion just before action 37154, which belong to the influence range. In Figure 6, a different anchor was chosen to show the dominance of the rewarding methods. In particular, we observe six events that cause negative fluctuations, mainly for Dynamic1 method. After the fourth event, Dynamic1 is overlapped by Dynamic reward3 and at the end by Dynamic reward2.

Finally, Figure 7 shows the results of the generated graph with 5000 nodes. Here, a split event take place at the beginning, which affects the performance of all methods. Nevertheless, not long after the third event and in combination with the activation of the static algorithm, Dynamic reward1 and reward2 have a significant improvement in recall and precision values, which has a consistently positive impact on the F1 score. On the other hand, Dynamic1 and Dynamic reward3 are almost congruent for a long time, but after action 218895 and until the end, the latter method performs much better. For the other two rewarding methods, it is obvious that recall and precision reflect the positive result of F1-Score.

The fluctuations we observe could be explained in two ways. First of all, many actions in the influence range of anchor are occurred. For instance, after a missing of an edge with extra reward the community coherence breaks, which reflects on quality metrics. Secondly, as mentioned previously, due to the fact that the static algorithm is used after a fixed batch of actions and not whenever the graph generator provides the ground truth communities, this have an effect on our outcome.

5 Conclusions

Dynamic local community detection constitutes a research field that has drawn scientists' interest the last years. In the present work, we focus on the discovery of local communities that contain important nodes termed anchors. Our aim is not only to identify such communities but also track their evolution over time as new edge insertions and/or deletions occur in a network. To achieve this, we suggest a multi-step framework that firstly applies a static algorithm to discover the initial anchor's community and then for each incoming edge change in the influence range of the anchor, we update the anchor's community. Influence range is used to minimize the avalanche effect. With a view to discover the most stable anchor's community, we suggest using a node rewarding method. That is, for each update, we suggest to reward the stable edges in the anchor's influence range by a weight increase. A preliminary experimental evaluation of the proposed framework is conducted using three different synthetic datasets. We also used three proposed rewarding methods and compared the results with the case where no rewarding method is used. Our findings indicate that all three dynamic methods with rewards (Dynamic reward1, Dynamic reward2 and Dynamic reward3) outperform the dynamic method without rewards in terms of recall, precision and F1 score.

This work contains preliminary results and we intend to extend these results along the following axis: 1. Extended experimental evaluation of more rewarding schemes that take into account the history of edges. 2. Experimentation on real temporal networks. 3. Elaborate tuning of the various parameter of the rewarding schemes and 4. Efficiency comparison between different rewarding schemes since the more complicated a scheme is the more time it needs per action.

Acknowledgment

"Georgia Baltsou is co-financed by Greece and the European Union (European Social Fund-ESF) through the Operational Programme "Human Resources Development, Education and Lifelong Learning" in the context of the project "Strengthening Human Resources Research Potential via Doctorate Research - 2nd Cycle" (MIS-5000432), implemented by the State Scholarships Foundation (IKY)."

"This research was supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "2nd Call for H.F.R.I. Research Projects to support Faculty Members & Researchers" (Project Number: 3480). "

References

1. Georgia Baltsou and Konstantinos Tsichlas. Dynamic community detection with anchors. 2022.
2. Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. In *International Conference on Ad-Hoc Networks and Wireless*, pages 346–359. Springer, 2011.
3. Jiyang Chen, Osmar R Zaiane, and Randy Goebel. Detecting communities in large networks by iterative local expansion. In *2009 International Conference on Computational Aspects of Social Networks*, pages 105–112. IEEE, 2009.
4. Daniel J DiTursi, Gaurav Ghosh, and Petko Bogdanov. Local community detection in dynamic networks. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 847–852. IEEE, 2017.
5. F1 score lemma. F1 score lemma — Wikipedia, the free encyclopedia, 2020.
6. Yang Gao, Hongli Zhang, and Yue Zhang. Overlapping community detection based on conductance optimization in large-scale networks. *Physica A: Statistical Mechanics and its Applications*, 522:69–79, 2019.
7. Kun Guo, Ling He, Jiangsheng Huang, Yuzhong Chen, and Bing Lin. A local dynamic community detection algorithm based on node contribution. In *CCF Conference on Computer Supported Cooperative Work and Social Computing*, pages 363–376. Springer, 2019.
8. Frank Havemann, Michael Heinz, Alexander Struck, and Jochen Gläser. Identification of overlapping communities and their hierarchy by locally calculating community-changing resolution levels. *Journal of Statistical Mechanics: Theory and Experiment*, 2011(01):P01023, 2011.
9. Vassilis Kostakos. Temporal graphs. *Physica A: Statistical Mechanics and its Applications*, 388(6):1007–1023, 2009.
10. Panagiotis Liakos, Katia Papakonstantinopoulou, Alexandros Ntoulas, and Alex Delis. Rapid detection of local communities in graph streams. *IEEE Transactions on Knowledge and Data Engineering*, 2020.
11. Feng Luo, James Z Wang, and Eric Promislow. Exploring local community structures in large networks. *Web Intelligence and Agent Systems: An International Journal*, 6(4):387–400, 2008.
12. Giulio Rossetti. Rdyn:graph benchmark handling community dynamics. *Journal of Complex Networks*, 5(6):893–912, 2017.
13. Giulio Rossetti and Rémy Cazabet. Community discovery in dynamic networks: a survey. *ACM Computing Surveys (CSUR)*, 51(2):1–37, 2018.

14. Mansoureh Takaffoli, Reihaneh Rabbany, and Osmar R Zaïane. Incremental local community identification in dynamic social networks. In *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*, pages 90–94. IEEE, 2013.
15. Yanhao Yang, Meng Wang, David Bindel, and Kun He. Streaming local community detection through approximate conductance. *arXiv e-prints*, pages arXiv–2110, 2021.
16. Anita Zakrzewska and David A Bader. A dynamic algorithm for local community detection in graphs. In *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 559–564. IEEE, 2015.
17. Anita Zakrzewska and David A Bader. Tracking local communities in streaming graphs with a dynamic algorithm. *Social Network Analysis and Mining*, 6(1):1–16, 2016.