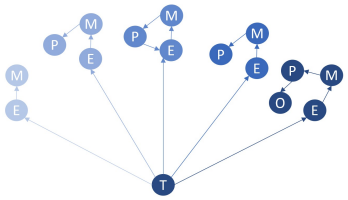
TEMPO

Management and Processing of Temporal Networks

H.F.R.I. Project No. 03480

D4.2 - Local Community Detection with Hints in Snapshot-Temporal Networks



Computer Engineering & Informatics Department
University of Patras
Greece
30/11/2025

D4.2 - Local Community Detection with Hints in Snapshot-Temporal Networks

Konstantinos Christopoulos and Kostas Tsichlas

November 30, 2025

Abstract

Community detection in dynamic networks is a challenging research problem. One of the main obstacles is the stability issues that arise during the evolution of communities. In dynamic networks, new communities may emerge and existing communities may disappear, grow, or shrink. As a result, a community can evolve into a completely different one, making it difficult to track its evolution (this is known as the drifting/identity problem). In this paper, we focused on the evolution of a single community. Our aim was to identify the community that contains a particularly important node, called the anchor, and to track its evolution over time. In this way, we circumvented the identity problem by allowing the anchor to define the core of the relevant community. We proposed a framework that tracks the evolution of the community defined by the anchor and verified its efficiency and effectiveness through experimental evaluation.

1 Introduction

Complex systems are very often represented by networks, since they can successfully demonstrate the natural structures and functions of various fields such as communication, biology, and the World Wide Web, where huge amounts of data are constantly being generated. Community detection, a fundamental task of network analysis, focuses on revealing group of nodes that are densely connected to each other and loosely connected to the nodes in the other groups in the network and has attracted the attention of many researchers. For decades, great efforts have been made to detect global communities, i.e., the partitioning of an entire network into communities [1]. However, there are many cases in which researchers are only interested in the communities of specific nodes. Therefore, in recent years, there has been an increased interest in exploring local communities based on a few query nodes [2, 3, 4]. From the computational cost point of view, the local community discovery problem is better suited to uncovering the community structure for nodes of interest in large networks, considering real-world scenarios such as purchase or social networks [5].

Most existing work on community detection, whether global or local, aims to uncover the community structure of static networks. Static networks have an unchanging structure over time. However, most real-world networks change over time. Such networks are called temporal or dynamic. A very recent survey of local community detection in both dynamic and static networks was presented in [6]. Furthermore, there are cases where relations between nodes are established only instantaneously and the structure of the networks changes rapidly, e.g., in email communication. The most common approach to handling such data networks is the (graph) streaming model. A graph stream consists of a sequence of updates on the edges of a graph. Time is defined in terms of the order of the updates within this sequence. The lifetime of an edge is defined as the time between its insertion and removal [7, 8], which in fact defines the notion of time with respect to these transactions (transaction time). Compare this notion to the valid time notion, where each edge update carries its valid interval, that is, the time during which the update is in effect.

In most local community detection approaches in the literature, communities are uncovered starting from seed nodes with particular topological importance according to a specific metric, such as node degree. As one can deduce, such nodes can be the most important according to the chosen measure, but this does not mean that they are always the right seeds for local community detection approaches. In the present work, we focused on local community detection of particular nodes in graph streams by extending the theoretical framework of [9]. More specifically, our goal was to

uncover the evolution of the community of a node (or a set of nodes) of particular interest, called the *anchor*. The anchor has a special meaning for its community. This importance may or may not be reflected in the topological properties of the nodes and is based on knowledge from the external environment of the network. This node defines the evolving community and acts as an anchor for the community, thus circumventing the identity problem.

As a toy example, one could imagine a football team in a social network. This community evolves as new fans join or existing fans may stop supporting the team. However, the core fans (e.g., the Ultras) of the team are more stable and in a way act as an anchor for this community. Imagine a first-class player of the team that has a large degree of centrality (much larger than the Ultras). This player is topologically central in the community referring to this team. However, when he decides to leave the team then all his connections within this community are severed but the community continues to exist due to its more permanent members (any of them can be chosen as an anchor). Another motivating example could be an IoT network. The IoT is defined as a network of connected devices and end systems that directly interact with each other to collect, share, and analyze important data via the cloud [10]. In such networks, the connections between nodes are not stable, but change over time. Thus, as an example, someone might be interested in uncovering the evolution of the community to which a particular switch device node belongs. The former node would act as an anchor for this community.

Another motivating example comes from social networking platforms. TikTok is an example of a network-based social platform, in which people every day express their opinions about many topics. A reaction of a user (node) to the challenge of another user constitutes an edge between both users in a reaction network model. In this case, the anchor node can be a person that issued such a challenge that may have a certain number of reactions. Applying local community detection with the anchor being the person that issued the challenge, we can discover not only the people that reacted to this challenge but also the people that reacted to such reactions and further see how this temporary community evolved over time from the time it started until the time people lost their interest. As a result, researchers can further analyze the dynamics of the community that, for example, may be used in the evolution of the sentiments within this community [11]. Additionally, sometimes such challenges may be characterized as dangerous, and discovering the evolution of the community may lead to the identification of evolutionary patterns that allow them to distinguish the communities of users who participate in dangerous and non-dangerous challenges [12].

Another motivating example is related to tracking potential COVID-19 cases emanating from a single infected person who recently traveled from overseas. This person could act as an anchor for identifying a community that potentially has been infected either directly or indirectly by her/him. We assume the existence of a contact temporal network. This could be constructed by using appropriate mobile apps (e.g., [13]). The identification of this community allows the application of preventive policies, such as suggesting that all people in this community should check for infection. However, we must highlight ethical issues related not only to the construction of such a contact network but also to the process of identifying the community. This is because the identification of the community may lead unaware people to be subjected to unnecessary actions and at the same time reveal personal information. The ethics of shared COVID-19 risks are discussed in [14], while issues related to shared responsibility in health policy can be found in [15].

Contributions

Our present work focused on identifying the community of a particular node, called an anchor, which is assumed to be of particular importance to that community due to external knowledge related to the network. To achieve this, we proposed a multi-stage framework in a graph streaming setting that updates the community whenever an edge is updated (inserted or deleted) "near" the anchor. We experimentally showed how promising the proposed framework is when compared to other methods in both synthetic and real datasets. Our contribution is twofold:

- From a **modeling perspective**, our contribution lies in introducing the notion of the anchor node in the local community detection problem into time-evolving networks.
- From an **algorithmic perspective**, a general multi-step framework was proposed that can be used to detect 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 in Section 5 and conclude in Section 6.

2 Related Work

Local community detection, also known as the seed set expansion problem, has attracted the attention of researchers because only a small part of the network is processed. This is either because the network is so large that it is impossible to look for all communities or because the user is interested only in communities in a small part of the network. A striking characteristic of local community detection is that the choice of the seed(s) defines the detected local community that can be quite different from the one that the seed would belong to in the case of global community detection. Many different approaches have been proposed for this problem. However, the literature on dynamic networks and especially on graph streams is much smaller than that on static networks. In the following, we discuss algorithms for local community detection in a streaming setting that are closely related to our work.

Ref. [16] used an online approach to find communities in data streams, where at each given moment they maintain the current graph of interactions in main memory and store previous graphs of interactions on the disk. In [17], an incremental method, to update the communities of a graph segment when a new incoming graph is added, was proposed. Random walks with restart were used and the suggested approach requires the maintenance of the whole graph structure. In addition, the authors of [18] adopted the static L-metric approach [19] to find dynamic communities in an incremental way. The L-metric is a measure expressing the observation that a community has a lower number of edges to other communities than to nodes within the community. At each snapshot, communities are discovered using information from previous snapshots. At the end, communities found in different snapshots are matched based on their similarity (L-metric). Experiments have shown that this method leads to meaningful communities. Additionally, in [20], the authors studied the problem of community detection in a streaming environment where the rows of the graph's adjacency matrix are revealed one by one, as they believe that maintaining the entire graph is prohibitively expensive. They proposed an online algorithm with a space complexity that grows sub-linearly with the size of the graph. Furthermore, in [21, 22], a dynamic method of expanding the seed set was proposed, where the authors proposed to incrementally update the fitness score of each snapshot. To center the community around the seed, their method ensures that the order of fitness scores remains monotonically increasing by tracking the order of added nodes. Experiments have shown that the proposed method is quite fast and the performance is better when low latency updates are required. The authors of [23] suggested an incremental community detection approach for high-volume graph streams based on a batch-oriented algorithm named DEMON [24]. The suggested algorithm considers only adding edges in an incremental way and requires maintaining the entire graph structure. Moreover, in [25], a method called PHASR was proposed to find the temporal community with the lowest conductance. This work aimed to find communities with stable membership over time. Experiments have shown that the proposed method has a low running time and manages to find high quality communities. In [26], a metric called local fitness was used to first find the starting nodes of a community and run a static algorithm to define the communities in the first snapshot. In subsequent snapshots, they used a metric for node contribution to gradually reveal the communities. Their experiments showed that the proposed method reveals communities with high accuracy. Despite the fact that the work in [27] pertains to static networks, we mention the importance of this research while the authors deal with the problem of the maximal a- quasi-clique; from a local community perspective, detecting the communities of a specific node of interest. Furthermore, the authors of [28] proposed the HqsMLCD algorithm in order to detect multiple overlapping communities for a given starting node. The notion of high quality seeds was introduced, which are obtained by the embedded candidate subgraph. Their motivation was to define a local community detection method that is sensitive to the local structure of the seed node. Experiments in real datasets showed that the aforementioned approach detects high quality communities. Finally, a method named CoEuS [29] has been proposed for local community detection in graph streams. The method works in a rather restricted setting where only a single access to the stream is assumed and

the working memory is limited. Experiments with networks have shown that the algorithm is able to discover local communities with high accuracy. In contrast to existing work on local community detection in a streaming setting, our proposed multilevel framework focuses on community detection of a particular important node, called anchor. This work is the first to propose the notion of "anchor" in graph streams.

3 Problem Formulation and Methodology

In this section, we introduce some basic terms and state our research problem more formally. Then, we discuss the framework for local community detection with anchors in stream graphs.

3.1 Preliminaries

The network is denoted as $G = (V, E_t)$ and consists of a static node set $V = \{1, ..., n\}$, where $n = |V|$ is the number of nodes, and $E_t \subseteq V^2$ is the set of edges at time t. Given the definition of network G, a community can be defined as a subset of nodes in G such that the density of connections within the subset is greater than the density of connections between the subset and the rest of the network G. That is to say, a community can be thought of as a group of nodes that have stronger relationships with each other than with nodes in other parts of the network.

In this work, we considered a streaming model of computation on the graph in the sense that edges are processed in a continuous and incremental manner, as they arrive in a stream, instead of processing the entire graph structure in bulk. Thus, let t be a discrete time domain with time steps $1, 2, \ldots$. A streaming source for edges in G can be defined as a function $s : t \rightarrow P(V^2)$, where $s(t) = E_t$ represents the set of edges in the graph at time t, and $\delta s(t) = \{e(t)\}$ represents the single edge update at time t, with $e(t)$ being an edge in the graph. In other words, the streaming source $s(t)$ generates one edge update $\delta s(t)$ per time unit in the graph G. The update can be an addition, a deletion, or a modification of an edge in V^2. The result of this update on the stream graph is that the communities of the corresponding network may change. Dynamic community detection is the process by which we can observe the evolution of communities in a network subject to such edge updates.

A *Local Community* (LC) is defined as the community to which the seed nodes belong. Seed nodes are the nodes that define the community to be discovered. Thus, a network G can be divided into LC and the rest of the network $G - LC = U$. Figure 1 shows the commonly accepted definition of the local community in a network G. Based on Figure 1, we can define three types of edges of LC: internal, boundary and external. Internal edges are the edges between nodes in LC. Boundary edges are those between nodes in LC and U. Lastly, edges between nodes in U, are called external. We also define the internal and boundary degree of LC as the number of internal and boundary edges of LC, respectively. In Table 1 we summarize the notation used throughout the paper.

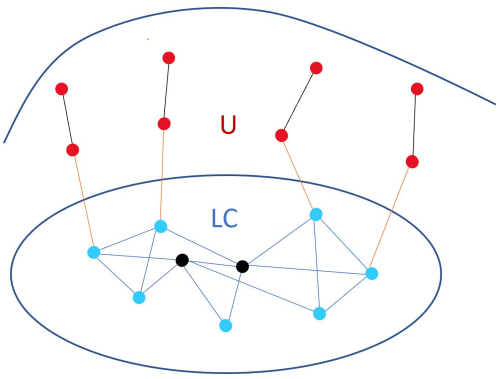


Figure 1: The shape of a community. Black nodes are the seed nodes. Blue nodes are nodes within the community while red nodes are outside of the community.

Table 1: List of basic symbols and abbreviations used in the present work.

Symbol/Abbreviation	Description
G_t	The network at time instance t—if no time index is given (G) then time is irrelevant
V	The node set of G
N	The total number of nodes in G ($= \|V\|$)
E_t	The edge set of network G at time t
$w(e)$	Weight of edge e
A	The anchor node
C_t	The community of the anchor at time t
$k_{in}^{C_i}$	The total internal degree of LC, when the i-th node is inserted to the community
$k_{out}^{C_i}$	The total boundary degree of LC, when the i-th node is inserted to the community C
$N(u)$	The set of neighbors of node u
R	The radius of the ball centered around the anchor A
A_R	The set of nodes, called influence range, in distance at most R from anchor A.
$s(t)$	The streaming source
b	The size of the batch in the streaming algorithm
LCDS-A	The proposed General framework of Local Community detection in graph streams with Anchors
DWR	The Dynamic With Rewards algorithm. It is an instance of LCDS-A for a particular reward scheme and quality metric.
DOR	The Dynamic WithOut Rewards algorithm [22]
SWR	The Static With Rewards algorithm
SOR	The Static withOut Rewards algorithm

3.2 Problem Formulation

In the following, we discuss more formally the problem of local community detection in graph streams with anchors. Given a node A called the anchor, the network G and the streaming source $s(t)$, our aim is to discover the community C which includes A. As the network evolves, the community C may also change. Our goal is to uncover the community of the anchor during the evolution process. We assume that the anchor is of particular importance to the community to which it belongs, due to external knowledge. Thus, it acts as a reference point for this community, i.e., the anchor in a sense defines the community to which it belongs.

To minimize the avalanche effect (the avalanche effect corresponds to the phenomenon where communities can experience significant deviations compared to what a static algorithm would compute in each instance of time) [30], we proposed to limit the community update only to a region of influence around the anchor. The influence range A_R of anchor A, is the set of nodes in the ball of radius R with the anchor A being the center of the ball. This means that within the influence range all nodes with shortest paths to the anchor A of length $\leq R$ are included. For example, A_1 contains the anchor as well as all its adjacent nodes. In general, a high value for the influence range would increase the requirements for the process, as a larger network area is explored. Internal, boundary and external edges with respect to the influence range are similarly defined to the case of the community.

Moreover, to identify the most stable anchor community, we applied a node reward method. That is, for each update, we rewarded the edges in the anchor's influence range by increasing their weight. For example, in the case where $R = 1$, all edges of the anchor as well as all edges between nodes in $N(A)$ are rewarded a weight in order to strengthen the ties of the anchor to its adjacent nodes. This procedure leads to the identification of a community that is "more" centered around the anchor. In the case of A_1, we may choose a very simple rewarding scheme that simply sets the weights of the edges of the anchor to be equal to a constant larger than 1. For instance, in case we choose this reward to be 2 then all incident edges of the anchor get a weight equal to 2 (recall

that graph is unweighted, that is, it is assumed that all edges have weight 1). In addition, all the edges between nodes that are adjacent to the anchor receive the same reward, creating in this way weighted triangles around the anchor. An example of this process is shown in Figure 2.

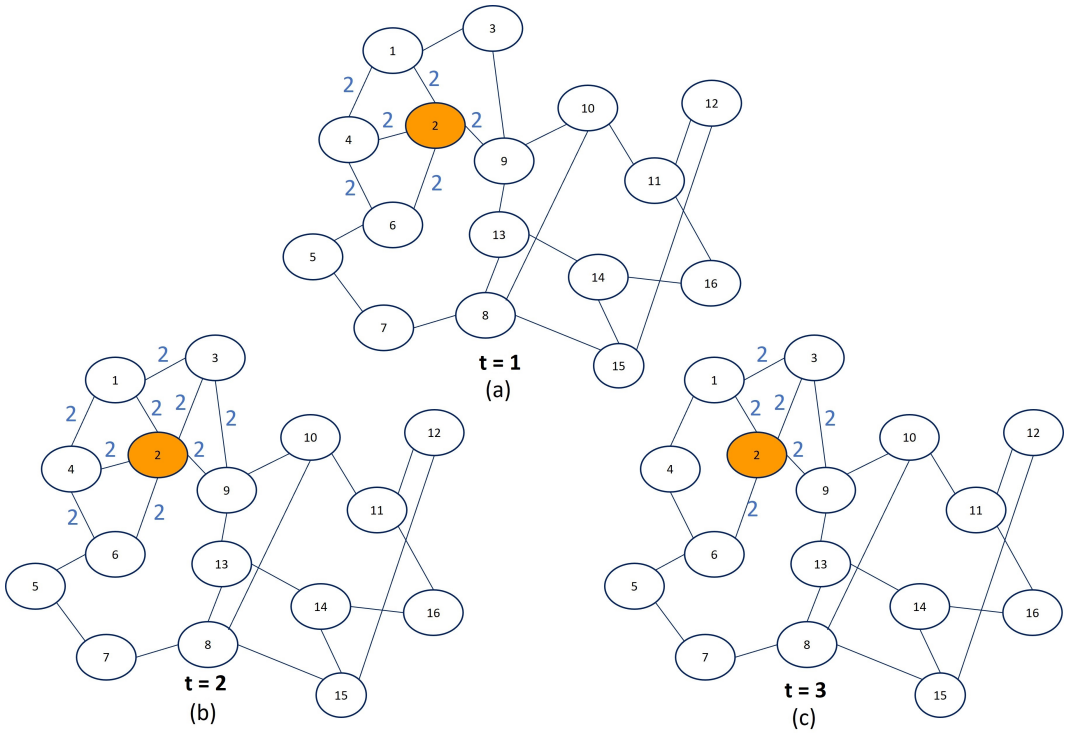


Figure 2: Applying rewards for local community detection with anchors in case $R = 1$: (**a**) The initial rewards around anchor is 2. (**b**) The edge $(2, 3)$ appears in the streaming source within the influence range of the anchor. The rewards are updated accordingly. (**c**) The deletion of edge $(2, 4)$ appears in the streaming source within the influence range of the anchor and the rewards are updated accordingly.

Our proposed framework uses a quality metric to guide the incremental construction of the anchor's community. The one we chose based on its simplicity and performance was the f_{monc}, which is defined as the ratio of intra-community edges to all edges with at least one endpoint in community C [31]:

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

where k_{in}^{C} and k_{out}^{C} are the internal and boundary degree of community C, and α is a positive real-valued parameter, controlling the size of the communities, i.e., lower α values allow larger community sizes.

A part of the proposed algorithm uses a static algorithm as suggested in [22], (see Algorithm 1). This greedy algorithm searches for new community members in the neighborhood of the current community. Each time, utilizing the corresponding quality measure, the fitness score f^{C_i} is estimated using the internal and boundary degree of the community. The node that produces the largest increase in the fitness score is chosen for addition in the community. When a new node u is added to C_{i-1}, a new community C_i is assigned to the node u_i, for $i > 0$, and as a consequence, regarding the size of the new community it holds that $|C_i| = |C_{i-1}| + 1$. For each node u_i, the pointer i declares the order in which the node u was added to the community. For example, C_0 is the community that contains only the anchor A with fitness score f^{C_0}, while C_1 is the community that results from the addition of the chosen node u_1 to community C_0. Lastly, the corresponding fitness score will be f^{C_1}, with $f^{C_1} > f^{C_0}$. Table 2 depicts this process. The final community C identified through the preceding process is the community $C_n, n \geq 0$, such that no node can be added to C_n resulting in the

increase of its fitness score. This incremental construction of C defines a sequence of communities $C_0, C_1, \ldots, C = C_n$, termed *incremental community sequence* henceforth.

Algorithm 1 Static algorithm [22].

Input: $G(V, E), A$

```
C ← {A}
fitnessmax = 0
nodemax = 0
while add new nodes in C do
    for u ∈ N(C) do
        if fitness_score(C ∪ {u}) ≥ fitness_score(C) then
            fitnessmax ← fitness_score(C ∪ {u})
            nodemax ← u
        end if
    end for
    C ← C ∪ {nodemax}
end while
```

Table 2: The incremental community sequence.

Sequence of added nodes	A	u_1	u_2	...	u_n
Incremental Community Sequence	C_0	C_1	C_2	...	C_n
Internal edges of C_i	$k_{in}^{C_0}$	$k_{in}^{C_1}$	$k_{in}^{C_2}$	...	$k_{in}^{C_n}$
Boundary edges of C_i	$k_{out}^{C_0}$	$k_{out}^{C_1}$	$k_{out}^{C_2}$	...	$k_{out}^{C_n}$
Fitness scores in ascending order	f^{C_0}	f^{C_1}	f^{C_2}	...	f^{C_n}

3.3 Local Community Detection in Graph Streams with Anchors

In this section, we propose the Local Community Detection in graph Streams with Anchors (LCDS-A) framework. From a bird's eye view, this framework first applies a reward scheme in the influence range of the anchor and then Algorithm 1 is used. When an edge is inserted/deleted that falls in A_R or C, then the rewards and/or the fitness scores are updated accordingly. At the next step, we check if some nodes should be removed from C and the fitness scores are further updated. In the last step, we check if the batch b of the inserted/deleted edges is completed. If yes, then we scan the fitness score sequence and apply Algorithm 1, otherwise, the next stream update is processed. The basic (five) steps of LCDS-A, given an initial graph G_0, are shown in Figure 3. In case the initial graph G_0 is empty (no edges), the first two steps are irrelevant. In this case, the initial community contains only the anchor A at $t = 0$.

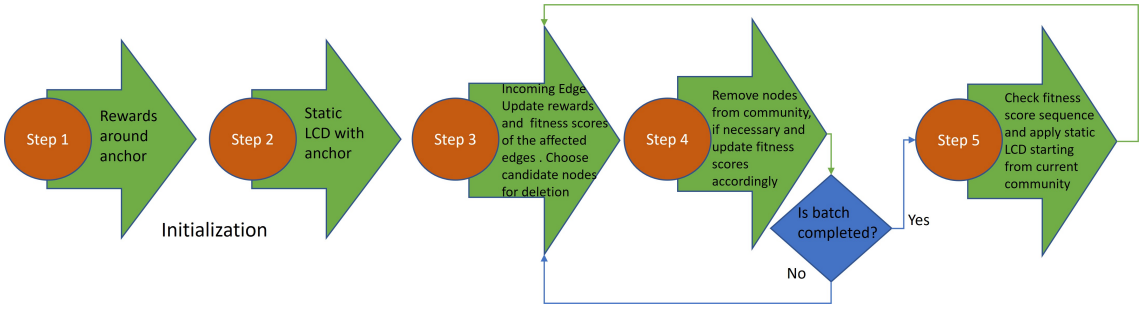


Figure 3: The general framework of the proposed approach LCDS-A for graph streams, using an initial graph G_0.

Since the anchor A is by definition of great importance to its community C, the goal of LCDS-A is to further strengthen its participation in C. In this way, we expected that local community

detection algorithms would provide better results. In the following, we make the convention for internal edges (u, v) in the community C, that u has been inserted in the incremental community sequence of C earlier than v based on their fitness scores. Similarly, for a boundary edge (u, v) of community C, we make the convention that $u \in C$ while $v \notin C$. The proposed framework is divided into two phases; the initialization phase and the streaming phase.

Initialization phase: Initially, in Step 1, we attach the rewards in the influence range A_R of the anchor by using a simple BFS traversal of all nodes at distance R. Then, in Step 2 we apply the static local community detection algorithm to find the community C that contains the anchor A. In case G is initially empty, community C consists only of the anchor while $K_{i,in}$, $K_{i,out}$ and f are initialized to 0. In this case, Steps 1 and 2 are omitted.

Streaming phase: In Step 3 of the framework, a stream update a is applied. This update can be either an insertion or a deletion of an edge. If a occurs in the anchor's influence range then: (1) the influence range must be recalculated and (2) the edge rewards are updated according to the reward method. If the updated rewards affect the internal and/or the boundary edges of the community, then the fitness scores in the incremental community sequence must be updated by recalculating the internal $k_{in}^{C_i}$ and boundary $k_{out}^{C_i}$ values used in the computation of the fitness score (see Section 3.2). We considered two cases based on how the weight of the affected edges has changed due to the stream update a. In the first case, the weight of the affected edge (u, v) is decreased. In this case, if the affected edge is internal, we keep node v in the list n_d (This is a list of candidate nodes for removal from C), becoming a candidate node for removal from community C. Node u is not selected as a candidate because after the weight decrease all $k_{out}^{C_i}$ will be decreased, from the moment where u was inserted in the incremental community sequence up until v is also inserted in this sequence. As a consequence, all the F1 scores will be increased and the ascending order of fitness scores in the incremental community sequence will be maintained up until v. Thus, node v is selected as a candidate because after weight decrease all $k_{in}^{C_i}$ will be decreased, from the moment where v is inserted in the incremental community sequence, meaning that the order of fitness scores of C_{i-1} and C_i may be violated (this means that in the sequence of nodes in increasing order based on the fitness score, if the candidate node was inserted in i-th order then the fitness score of C_{i-1} has become larger than the fitness score of C_i, which is a clear violation of the imposed increasing order in fitness scores), e.g., $f^{C_{i-1}} \geq f^{C_i}$, and needs to be checked. If the affected edge is boundary, then for the same reason as mentioned above, the increasing order of fitness scores will be maintained and we do not need to insert the internal node u in n_d list as a candidate node for removal from C.

In the second case, if the weight is increased, regardless of whether the affected edge (u, v) is internal or boundary, we insert node u in the list n_d, unless u is the anchor A, in which case nothing happens. u is selected as candidate node because after the weight increase in (u, v), from the time u was inserted in the incremental community sequence up until v was also inserted in the sequence (in case (u, v) is an internal edge), all $k_{out}^{C_i}$ will be increased. As a consequence, the sequence of fitness scores of C_{i-1} and C_i communities may be violated, e.g., $f^{C_{i-1}} \geq f^{C_i}$, and they need to be checked.

Irrespective of whether a occurs or not in the anchor's influence range, we should also check if a occurs in the community of the anchor and update the fitness scores of communities of the affected nodes, f^{C_i} for $i \geq 1$, since C may extend beyond the influence range. If a corresponds to a deletion of an internal edge (u, v) of C, we insert node v in the list n_d. If a corresponds to an insertion of an internal or boundary edge (u, v), we insert node u in the list n_d, unless u is the anchor A, in which case nothing happens. The way we select the candidate nodes for removal from C is exactly the same as mentioned previously in Step 3.

In Step 4, we check for each candidate node u_j in n_d whether the community C_j has a fitness score that violates the increasing order, that is $f^{C_{j-1}} \geq f^{C_j}$. In this case, the node u_j is removed from community C and C_j is also removed from the incremental community sequence. Then, we update the fitness scores of all succeeding communities f^{C_i}, for $i > j$. We insert in the list n_d all the neighbours of the removed node u_j that belong to community C and that have been inserted in the incremental community sequence later than u_j. We do not need to check neighbours that have been added earlier than u_j because they were added to C without taking into account u_j. The whole process is repeated until there are no neighbors affected by these changes and the list n_d is empty.

Finally, in Step 5, we check whether a batch of b stream updates has been processed. b counts the total number of stream updates that have occurred in the influence range of the anchor or in its community. Thus, stream updates that do not affect the community of the influence range are discarded and not counted within the batch. When a batch of stream updates has been completed,

we first check whether the incremental community sequence is valid, that is the fitness scores are in ascending order. If not, then we remove all nodes from this sequence from the leftmost violation to the end. For instance, given the sequence $f^{C_1} \leq f^{C_2} \leq f^{C_3} \geq f^{C_4} \leq f^{C_5}$, we observe a violation between $i = 3$ and $i = 4$, and thus all nodes from u_3 to u_5 should be removed from C. After the removal of nodes, the community C contains only three nodes, the anchor A, u_1 and u_2. Then Algorithm 1 is applied to the current community of the anchor, in order to add new node members in the community by extending the incremental community sequence. We must note at this point that the more frequent we run the static algorithm, the more accurate the result. However, the computational cost is also higher. Assuming that G_0 is empty (Steps 1 and 2 are not shown), the pseudo-code of LCDS-A is presented in Algorithm 2.

Algorithm 2 The pseudo-code of the proposed LCDS-A framework.

Input: $G(V, E_t), A, R, b$

$C \leftarrow \{A\}$
$A_R \leftarrow$ set of nodes within distance R from A
$k_{0,in} \leftarrow 0$
$k_{0,out} \leftarrow 0$
$f^{C_0} \leftarrow 0$
for $(u, v) \in E_t$ **do**
  Step 3:
  if $(u \in A_R) \vee (v \in A_R)$ **then**
    update A_R and $w(u, v)$ according to RW ▷ RW: ReWard scheme
    for $e = (u_I, v_I) \in C$ and $(u_I, v_I) \neq (u, v)$ **do**
      if $w(e)$ affected by RW **then**
        update $k_{i,in}$, $k_{i,out}$ and f^{C_i}
        if $w(e)$ is decreased and $v_I \in C$ **then**
          $n_d \leftarrow v_I$
        else if $w(e)$ is increased and $u_I \neq A$ **then**
          $n_d \leftarrow u_I$
        end if
      end if
    end for
  end if
  if $(u \in C)$ **then**
    if (u, v) is deleted and $(v \in C)$ **then**
      update $k_{i,in}$, $k_{i,out}$ and f^{C_i}
      $n_d \leftarrow v$
    else if (u, v) is inserted and $u \neq A$ **then**
      update $k_{i,in}$, $k_{i,out}$ and f^{C_i}
      $n_d \leftarrow u$
    end if
  end if

  Step 4:
  while $n_d \neq \emptyset$ **do**
    $u_j \leftarrow$ extract a node from n_d
    if $(f^{C_{j-1}} \geq f^{C_j})$ **then** ▷ f: the fitness score
      $C \leftarrow C - \{u_j\}$
      update $k_{i,in}$, $k_{i,out}$ and f^{C_i}
      for $v_\ell \in N(u_j)$ **do**
        if $(v_\ell \in C) \wedge (\ell > j)$ **then** ▷ Check if v_ℓ added to C later than u_j
          $n_d \leftarrow v_\ell$
        end if
      end for
    end if
  end while

  Step 5:
  if # of stream updates that belong to C or A_R is equal to b **then**
    Remove all nodes from the leftmost violation to the end in the incremental community sequence
  end if
  Run the static Algorithm 1 to add new nodes in C
end for

Time Complexity

The complexity of the DOR method, based on the analysis of [22], is $O(n^2 d)$, where d is the mean degree of the nodes in graph G and n is the size of the community of the anchor. Initially, we provide a very crude worst-case analysis of LCDS-A based on N and M, which are the number of nodes and edges of the graph respectively (M, n and d are quantities that change as the graph evolves—we have assumed that nodes do not change although many of them can have zero degree. However, asymptotic notation allows us to be more relaxed with these values assuming that they do not change much during a period of time. Indeed, we can safely assume that during a period of one batch the mean degree d as well as the number of edges M change only by a multiplicative constant. The size of the community n can only drop between two successive calls to the static algorithm (Step 5 of LCDS-A) and as such, n is an upper bound). DOR has a worst-case complexity of $O(N^3)$, since $d, n \leq N$. For each stream update (edge insertion/deletion) in the anchor's influence range/community, we update all rewards in time $O(M)$, since in the worst-case we will have to update the rewards of all edges in the graph. Then, we update the new fitness scores in time $O(N^2)$, since for each node in the community we need to calculate the internal and the boundary edges. For each update of a fitness score, if the node remains in the community then no other node is affected. If, on the other hand, the node is removed from the community then all its neighbors must have their fitness score recalculated. If a node is removed, then it is not inserted again in the community unless we are at Step 5. This implies an $O(N + M)$ step for these recalculations of fitness scores. This means that a crude upper bound of the complexity of each update that does not evoke Step 5, is $O(M + N^2)$. In case, Step 5 is evoked, then the cost of the update is increased by an additive $O(N^3)$ since the bottleneck in this step is the use of the DOR algorithm. Thus, in total, we get a complexity of $O(bN^2 + N^3)$ for a batch of b stream updates, since $M < N^2$.

The above analysis is very pessimistic since the size of the community is expected to be much less than the size of the graph ($n << N$) and the internal computation of the LCDS-A algorithm is also expected to be less intense than the one implied above. A better estimation (although still pessimistic) can be achieved by using more graph-related parameters in the time complexity of each update. To this end, as already stated, during a batch of b updates, Algorithm 1 is executed in $O(n^2 d)$ time. Moreover, the time we need in order to calculate the rewards of the affected edges in the influence range of the anchor A, in the worst case, is $O(bd^R)$. This is because during the b updates, the rewards will change in a ball of radius R around the anchor of size at most d^R—for $R > 1$ this is a clearly pessimistic upper bound since it assumes no common neighbors between nodes. In addition, the endpoints of some edges that belong to C will be affected and the corresponding fitness scores should be recalculated, as well. This can be achieved in $O(bd)$, since for each of the b stream updates, we need $O(d)$ time to recalculate the fitness scores of the neighbors of the affected nodes. Assuming that the average number of violations per update is 1 (This implies that during a batch, the community will lose b nodes. Of course, in the worst-case, in one stream update all nodes may be removed from the community. However, our experimental results imply that the average number of violations per stream update is lower than 1), then, for each node removal from C, d new neighbours of the removed node should be inserted in the n_d list, and for each one the fitness score will be estimated in $O(d)$. Thus, for d neighbours, the total time is $O(d^2)$, and for all b updates in a batch, the time we need is $O(bd^2)$. In total, the time complexity is $O(b(d^R + d^2) + n^2 d)$ given the plausible assumptions we made. This means that a rather pessimistic upper bound per stream update is $O\left(d^R + d^2 + \frac{n^2 d}{b}\right)$.

Regarding the other three baseline algorithms, the static without rewards scheme requires $O(bn^2 d)$ for a batch of stream updates, since the static algorithm is applied for each stream update. The static with rewards scheme requires $O(bd(n^2 + d^{R-1}))$, since besides the time required for applying the static algorithm, it requires $O(bd^R)$ additional time to calculate the rewards after each update. Finally, the dynamic without rewards requires $O(d(bd + n^2))$ with a reasoning similar to the analysis of LCDS-A. Apparently, LCDS-A is faster than the static baseline algorithms as it is expected but slower than the dynamic algorithm without rewards due to the maintenance of the rewards in the influence range.

4 Experiments

4.1 Experiment Design

In this section, we experimentally compared different baseline approaches with the proposed framework LCDS-A, on both synthetic and real datasets. The baseline approaches are (a) DOR (Dynamic withOut Reward), as proposed by Zakrzewska and Bader [22], (b) SOR (Static withOut Reward) and (c) SWR (Static With Reward). Regarding the approaches SOR and SWR, each time a stream update arrives, then Algorithm 1 is applied, incurring a rather large computational cost. In addition, for the SWR method, before we applied Algorithm 1, the weights of the edges in the influence range were recalculated.

LCDS-A is a general framework where we can apply various reward schemes and quality metrics. In our experiments, we used a specific instance of the general framework in terms of reward and quality metric. In order to differentiate this instance from the framework LCDS-A, we used the term DWR (Dynamic With Reward). First, we experimented with synthetic datasets. We used a representative node as an anchor A for each experiment with the synthetic datasets, which was determined by its degree. We experimented with different anchors of low, medium, and high degree when compared to the average degree of the network when the streaming process was ended (the last instance of the graph). Thus, a high degree node is a node that has degree considerably larger than the average degree at the last instance. Second, with respect to the real datasets, we gave an average F1 score for several important nodes and, consequently, certain anchor nodes A were chosen based on external knowledge about each dataset.

In the experimental evaluation of DWR, we assumed that the radius of the influence range was $R = 1$. We chose this value as a good compromise between efficiency and effectiveness. The anchor A was a single node and the reward scheme was very simple since we only assigned a chosen reward to all edges in A_1.

Finally, regarding the size b of the batch of stream updates, after extensive experimental evaluation we concluded that b should be analogous to the current size of the community or influence range. More precisely, the static algorithm was executed only when the size of stream updates b that belong to A_R or C were greater than a percentage of the current size of C or A_R. In this way, we gave an advantage to the community that suddenly loses a lot of members after a stream update.

4.2 Datasets

4.2.1 Synthetic Datasets

The synthetic datasets we used in our experiments were generated by RDyn [32], which is capable of generating dynamic networks that respect known real-world network properties, along with time-dependent ground truth communities with adjustable quality. Both merging and splitting of communities is allowed. The generator contains two important user-defined parameters. The first is the number of nodes of the generated dynamic network and the second is the number of iterations. Each iteration consists of a batch of stream updates (insertion/deletion of edges) and the number of these updates is not necessary the same for each iteration. For our experiments, we used only the anchor as the initial graph, followed by a full streaming procedure. Moreover, we used three different datasets generated by the RDyn generator. The basic properties of these datasets are described in Table 3.

Table 3: The basic properties of the synthetic datasets used.

Dataset	Nodes	Mean Degree	Iterations	Final Edges	Stream Updates
$SD1$	500	64	1000	1680	41,433
$SD2$	1000	54	1000	6226	50,871
$SD3$	5000	55	1000	25,590	251,107

4.2.2 Real Datasets

The real datasets we used for our experiments are datasets that can provide us with side information in the form of metadata about the meaning of certain entities. More specifically, the first dataset

we used in our experiments [33] consists of nodes representing employees of a company, while edges represent email communications between them. As side information, we already knew to which department each employee belongs and what position they hold there. We were interested in understanding the communication patterns of a specific node, e.g., a manager, based on the company email communication. We could apply local community detection for that node (manager), as the seed node, to identify the different groups of people she communicates with most frequently. This could reveal which departments or teams the manager interacts with most often, as well as any individuals who may be particularly influential or well-connected within those groups.

For example, we might find that the manager is part of a local community that includes other managers and executives within the company, as well as members of their direct team. Alternatively, we might discover that the manager is more closely connected to employees in certain departments, indicating that they may have a more specialized or targeted role within the organization. Based on these insights, we could develop strategies to improve communication and collaboration within the company, such as encouraging the manager to build stronger relationships with certain individuals or groups, or facilitating more cross-functional collaboration between departments or teams.

The second real-world dataset we used in our experiments is a terrorism dataset [34]. More specifically, the nodes represent terrorists and the edges between them represent attacks involving both endpoint nodes of each edge. As side information, we obtained the position of each terrorist in a terrorist group, e.g., the bomber, and the possible relationships between them. For instance, in the context of a dataset that describes terrorism in the last decades, if we chose a specific node such as "bomber" and applied local community detection, it could help us identify the different groups and networks that the bombers belong to, as well as the key players and influencers within those groups. This could potentially reveal important insights about the organization, structure, and tactics of terrorist groups. For example, by identifying the local communities that include bombers, we may be able to determine which groups are most active in carrying out bombings and whether there are any patterns or trends in their activities (such as targeting specific locations or types of targets).

Additionally, by analyzing the connections between bombers and other members of their communities, we may be able to identify potential collaborators, recruiters, or facilitators, which could be useful for law enforcement and counter-terrorism efforts. The basic properties of these datasets are described in Table 4.

Table 4: The basic properties of the real datasets.

Dataset	**Nodes**	**Actions**
terrorism	271	756
$email - Eu - core$	1006	16,706

4.3 Evaluation Metrics

To evaluate our proposed framework, we compared the results of our community detection with the ground truth communities generated by the synthetic dataset generator, in different time instances. However, a valid argument (to some extent) against using the synthetic generator's ground truth communities is that the detected community is influenced by the anchor. To this end, on the one hand, we tried to set up the generator so that the communities were not as intertwined, and on the other hand, we were more interested in comparing the methods to each other than in looking at the values of the metrics compared to the ground truth. Regarding the real networks, we compared our results, in each timestamp, with the same ground truth community as given from our dataset. The evaluation metrics that we used were the standard precision, recall, and F1 score [35]. 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. The harmonic mean is used instead of the simple average because this way the extreme values are penalised. The F1 score has been used extensively in the context of clustering and community detection. This is achieved by comparing to the ground truth communities, to measure how well the algorithm has grouped together items that share common characteristics or features. The F1 score is utilized as an evaluation metric in a lot of research regarding local and global community detection algorithms [36, 37, 38, 28, 39, 40] and has also been used in large related

surveys [30] as a low computational cost evaluation metric. The Jaccard coefficient [41] is another metric that could also be used in order to evaluate the detected communities. However, since the F1 score is linearly related [42] to the Jaccard coefficient, we chose the former for simplicity in presenting the results.

4.4 Experiment Results

4.4.1 Experiments on Synthetic Datasets

Our experiments were conducted on an Intel Core 2,9 GHz i7 processor with 16 GB memory. In addition, we used Python to implement the methods and we made use of the NetworkX, igraph, and numpy libraries. To begin with, for all datasets, experiments were conducted using several nodes (low, average and high-degree) and we present the most representative results. Regarding the figures, the values on the x-axis represent the number of iterations. The graph generator returns the graph partition after one or more iterations, and in each iteration the number of stream updates is not the same. As a result, each interval on the x-axis consists of the same number of iterations but a different number of stream updates. In the first experiment, we compare DWR (Dynamic With Rewards), with DOR using several nodes and six different rewards ($1.2, 1.5, 2, 3, 5$ and 10). In each experiment, the static algorithm was activated when the size of the batch of stream updates b that occur in the influence range or the community C was greater than $b = 30, 50, 70$ or 90 percent of their current size.

Analyzing the experimental results of SD1 dataset, in Table 5, we observe that all six rewards outperform DOR [22]. More precisely, while we increase the reward, we observe that the F1 score is increased as well, and the maximum difference is achieved when the reward is equal to 3. On the other hand, when the reward value is greater than 3, we see that the average F1 score is decreased. This happens because the weight difference between the influence range area and the rest of the graph is too high and as a result, when the static algorithm runs, it does not add any new nodes to the community. In addition, with respect to the batch size b, we do not observe any significant changes in the results. We realise some fluctuations regarding the rewards and a slight increase for DOR until batch is equal to 70%.

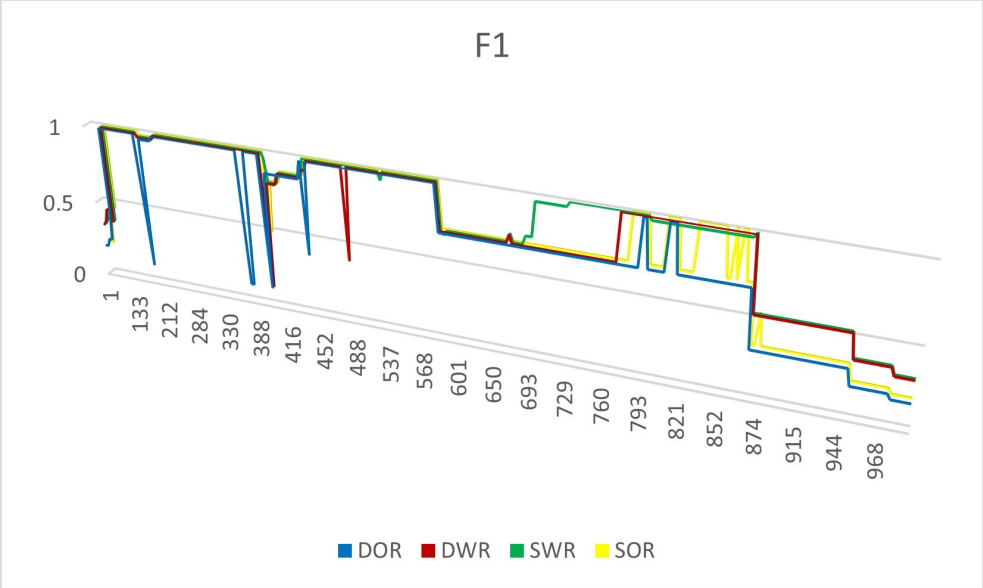
Having shown that DWR is better than DOR as far as F1-score is concerned, in Table 5 we also report results concerning execution time. In the third column, we represent the average execution time of DOR for each batch. Moreover, the last column shows the average execution time of DWR with multiple rewards for each batch. It is obvious that in both methods, while the batches are increasing the execution time is decreasing. This occurs because when the batch is low, then Algorithm 1 is called more times and this is time consuming. Comparing the third with the last column of Table 5, we observe that the average execution time of DWR with multiple rewards is slightly higher than DOR, and the percentage difference between these two methods is almost 13.5%. However, the F1-score results of DWR is almost 5.5% better in average than DOR and for certain reward and batch choices, the results of DWR are much better than DOR. Similar experiments were conducted for the next datasets and for this reason we used a batch equal to 30%, since it is the value that provides the best results.

Table 5: Average F1 scores and execution time of SD1 for multiple rewards ($1.2, 1.5, 2, 3, 5$ and 10) and batches. For the batch, we provide only the percentage.

	DOR	**Time**	**1.2**	**1.5**	**2**	**3**	**5**	**10**	**Time**
$b = 30\%$	0.709	37.2 s	0.736	0.748	0.764	0.777	0.767	0.733	44.7 s
$b = 50\%$	0.71	32.8 s	0.73	0.742	0.76	0.776	0.767	0.73	39.5 s
$b = 70\%$	0.714	31.9 s	0.725	0.741	0.761	0.771	0.76	0.731	35.6 s
$b = 90\%$	0.705	30.8 s	0.723	0.741	0.755	0.767	0.756	0.725	33.6 s

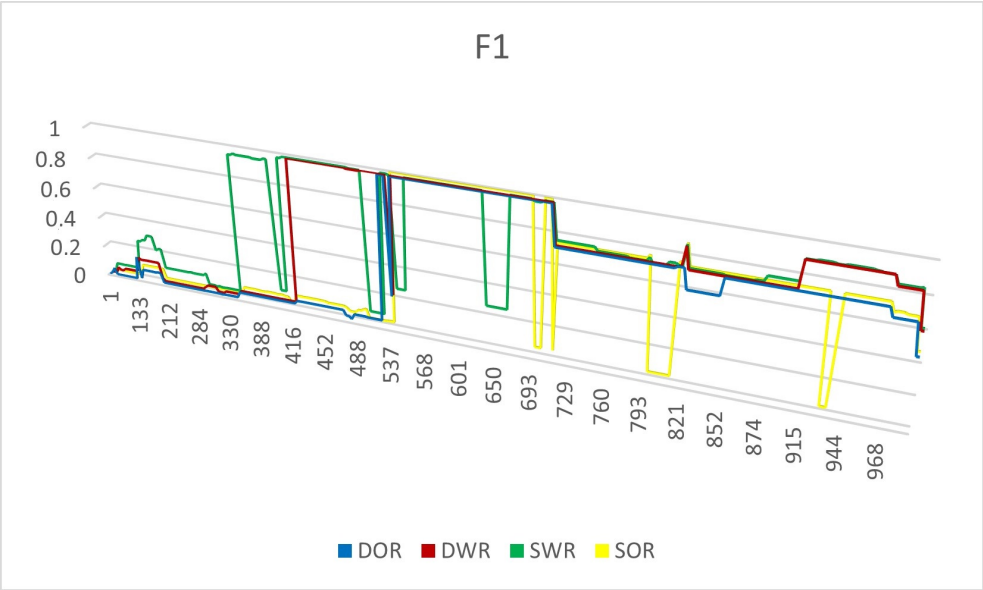
For the second synthetic dataset, SD2, we used a reward equal to 2, because for values greater than 2 our F1 score was decreased, and we utilized the static algorithm when b is 30%. Figures 4

and 5 present our results using nodes with high and average-degree respectively as anchors. In this experiment, we compared DWR with DOR and two baselines, SWR and SOR. The F1 score in Figure 4 exemplifies the superiority of our method. The average F1 score shows that using rewards is better by an additive 8% when compared to DOR. In addition, SWR is better than SOR by 8% as well. It is remarkable that our method outperforms SOR, and this result shows the significant outcome we can achieve by utilizing rewards.



	DOR	DWR	SWR	SOR
$b = 30\%$	0.731	0.814	0.845	0.764

Figure 4: Using a node with high degree as an anchor in the synthetic dataset SD2. The tuple at the bottom presents the average F1 score for the different methods.



	DOR	DWR	SWR	SOR
$b = 30\%$	0.512	0.652	0.677	0.465

Figure 5: Using a node with medium degree as an anchor in the synthetic dataset SD2. The tuple at the bottom presents the average F1 score for the different methods.

Looking in more detail at the results of recall and precision in Figures 6 and 7, respectively, we realise that the fluctuations in F1 score are due to the low recall. More precisely, the DOR algorithm, on average, has low recall because the detected community contains fewer nodes than those from the ground truth. Conversely, when we utilized rewards with either the Dynamic or the Static algorithm, the detected community contains an average of 81% of nodes of the ground truth community. In addition, we see that precision is almost equal to 1 for all methods, after the first few iterations. In Figure 5, working with an average-degree node, we observe a lot of ups and downs in the F1 score in all methods, except for in the proposed one. This happens because in the first third of the graph stream, the recall for DOR and SOR is very low and, at the same time, precision fluctuates wildly. At the beginning of the graph stream, until iteration 541, only a few edges are connected to the anchor and for this reason the values of the evaluation metrics are low, except SWR and DWR, which give better results from iterations 335 and 411, respectively. To summarize, the proposed method, as well as the static method with reward, both outperform the other two because on average the recall value is much better.

In the third synthetic dataset, SD3, (Figure 8) a low-degree node was used and, for this reason, until iteration 600, we get low F1 scores because there are a few edges connected to the anchor. Beyond iteration 600, we observe a significant increase in the F1 score for almost all methods, except SOR, and at the end DWR again outperforms the DOR method. At this point we need to make two observations. First of all, after iteration 600, our proposed method reaches an F1 score of nearly 98%, which is much better than the other three approaches. Secondly, the average F1 score for the SOR method is only 28.5%, and this occurs because without reward and with the combination of many edge deletions, the SWR method cannot reach the performance of the other three methods. Looking more carefully at the Figures 8 and 9, we observe the differences when we use nodes that have few and many connections respectively, from the beginning of the experiment. For instance, in Figure 9, the F1 score is quite high for all methods. The observed fluctuations for DOR, and especially for SOR, occur due to the extremely low recall. In these two methods, communities suddenly lose a lot of members and, as a consequence, recall is decreased. Regarding the precision of the DWR method, there are a lot of fluctuations in the first half of the stream updates. On the other hand, in the second half, precision is consistently higher than 90% and for this reason F1 shows a spectacular improvement. The most remarkable result in all the above experiments is that the proposed method outperforms the other three. Last but not least, on average, the SOR method

produces the worst results in recall in all the experiments. As a consequence, the F1 score is low and, comparing it with the SWR method, it gives us an obvious explanation about the prominence of the reward method.

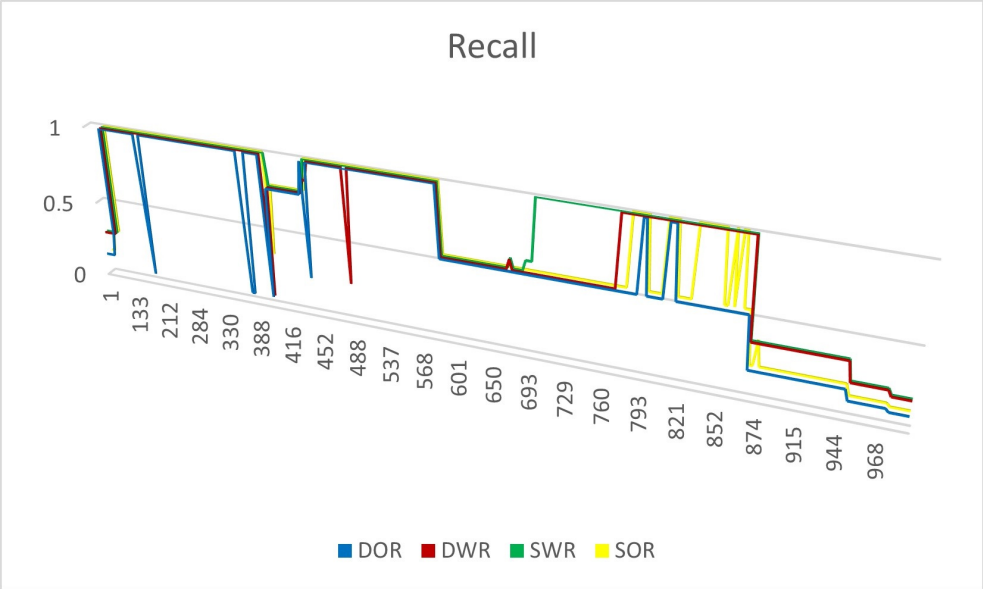


Figure 6: Using a node with high degree as an anchor in the synthetic dataset SD2.

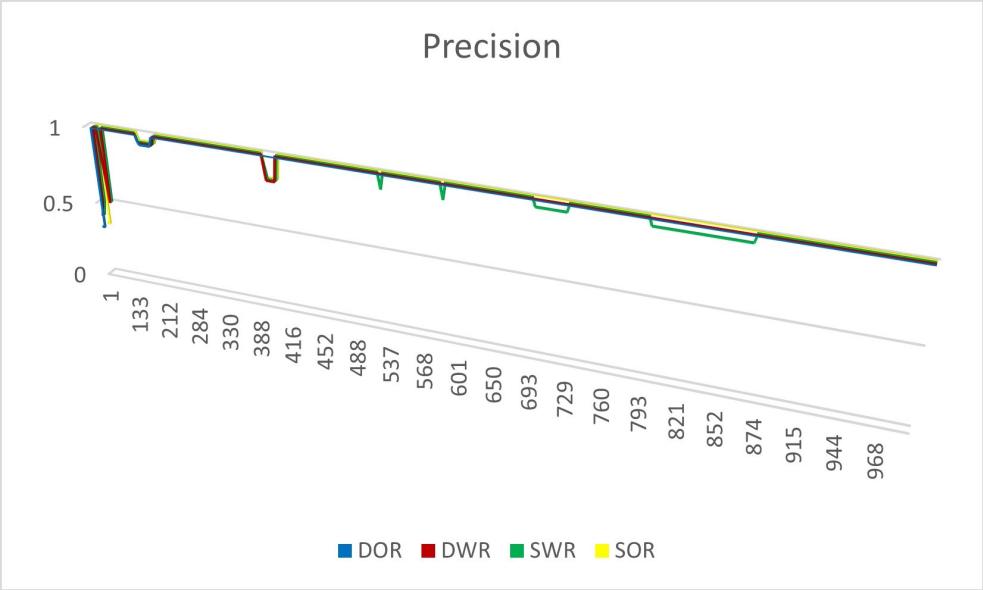
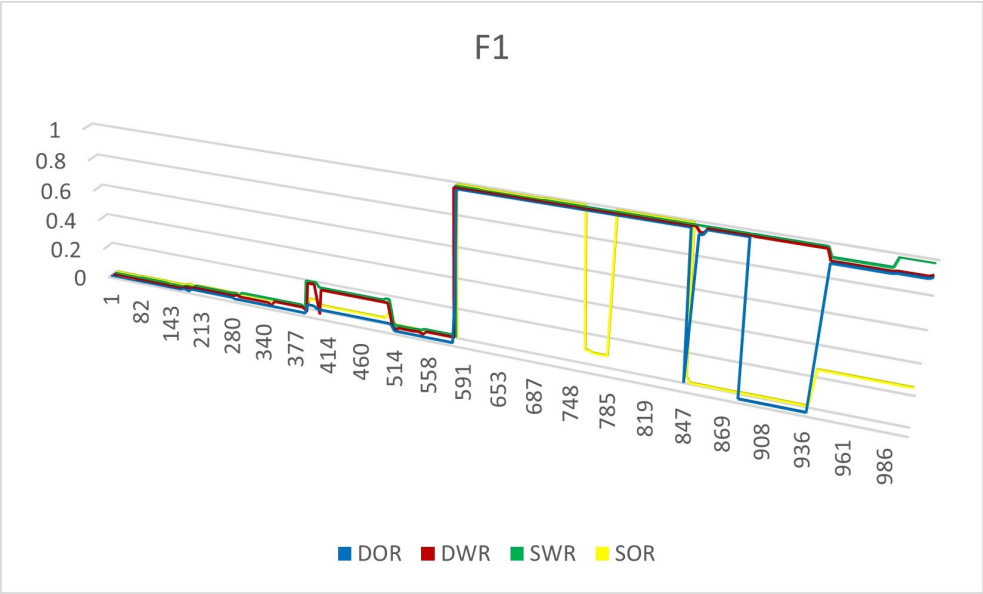
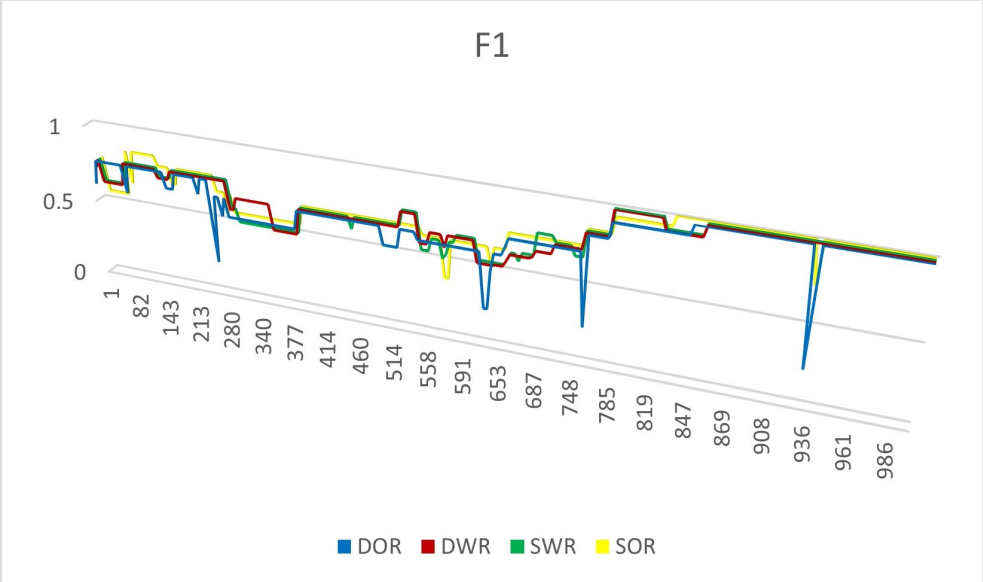


Figure 7: Using a node with high degree as an anchor in the synthetic dataset SD2.



	DOR	DWR	SWR	SOR
$b = 30\%$	0.441	0.541	0.546	0.285

Figure 8: Using a node with low degree as an anchor in the synthetic dataset SD3. The tuple at the bottom presents the average F1 score for the different methods.



	DOR	DWR	SWR	SOR
$b = 30\%$	0.765	0.783	0.774	0.774

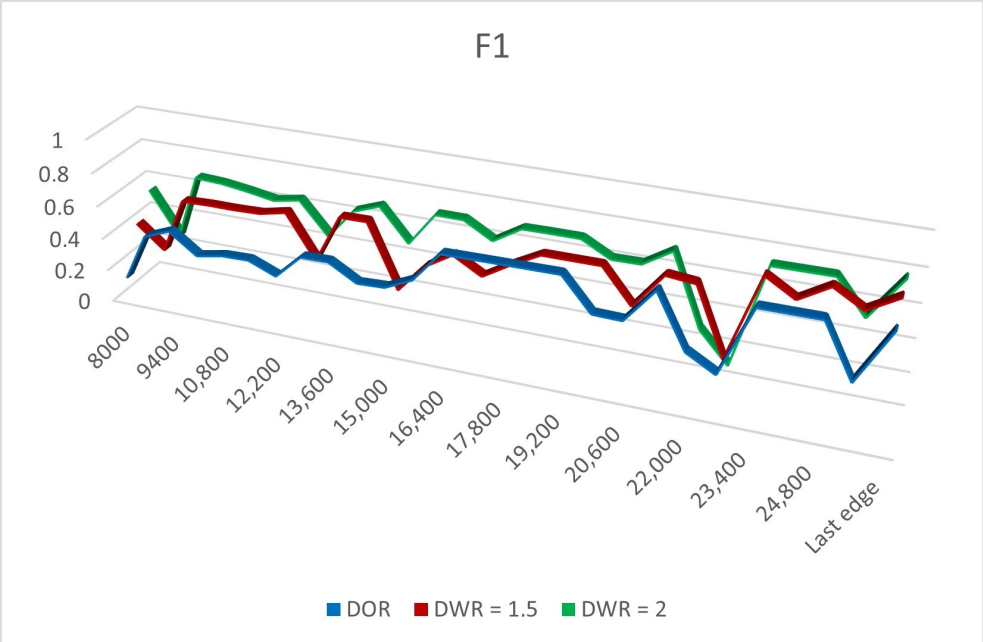
Figure 9: Using a node with high degree as an anchor in the synthetic dataset SD3. The tuple at the bottom presents the average F1 score for the different methods.

4.4.2 Experiments on Real Datasets

The experiments on real datasets using our proposed framework are quite promising. For these datasets, as stream updates we considered only insertions of edges. Moreover, both real datasets contain multiple edges and, in addition, the email dataset is directed. For our experiments, we pre-processed these datasets in order to remove multiple edges and make all edges undirected.

In this case, we have two types of experiments. First, we compared the DWR with the DOW method, for rewards 1.5 and 2 and for several different nodes. We did not use other rewards, as in synthetic datasets, because for rewards greater than 2 we did not observe significant changes. Second, we compared four different approaches to include as many alternatives as possible to identify the community of anchors. These approaches are: (1) SOR, (2) SWR, (3) DOR, and (4) DWR, which is our proposed method. We present the results of all these approaches in terms of the F1 score from experiments with real datasets considering different anchor nodes. When metadata about specific nodes was given, we used these nodes as anchors. In addition to these nodes, we also used low-degree nodes, average-degree nodes, and high-degree nodes in each dataset to test the behavior of our framework. Regarding the figures, the values on the x-axis represent the number of stream updates (inserted edges) while the y-axis corresponds to the quality metric used for the detected community when compared to the ground truth.

In the first experiment (Figure 10), using the email-EU-core network, we compared our proposed method with DOR and we applied the static algorithm when the batch was 30% of the current size of the community or influence range area. In this case, we gave the average of several nodes and for rewards equal to 1.5 and 2. As we can see, we achieved the best F1 score results when the reward was equal to 2. There is more than a 10% difference between DOR and DWR. Based on the analysis of the email-EU-core network, the chosen nodes for the experiments are the high, average, and low degree nodes.

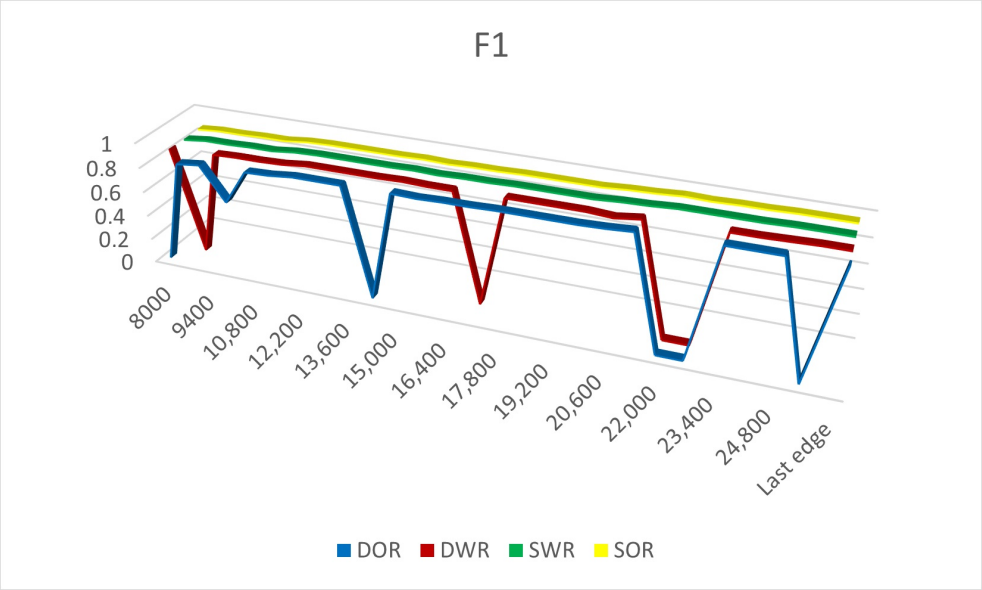


	DOR	1.5	2
$b = 30\%$	0.493	0.57	0.60

Figure 10: Average F1 score of several nodes as anchors, for high, average and low node degree, in the email dataset. The tuple present the average F1 score of the Dynamic, the Dynamic with reward 1.5 and the Dynamic with reward 2.

In Figure 11, the anchor is retrieved from metadata [43], which is the manager of a department that is of high importance. On average, DWR outperforms DOR and in addition, both static

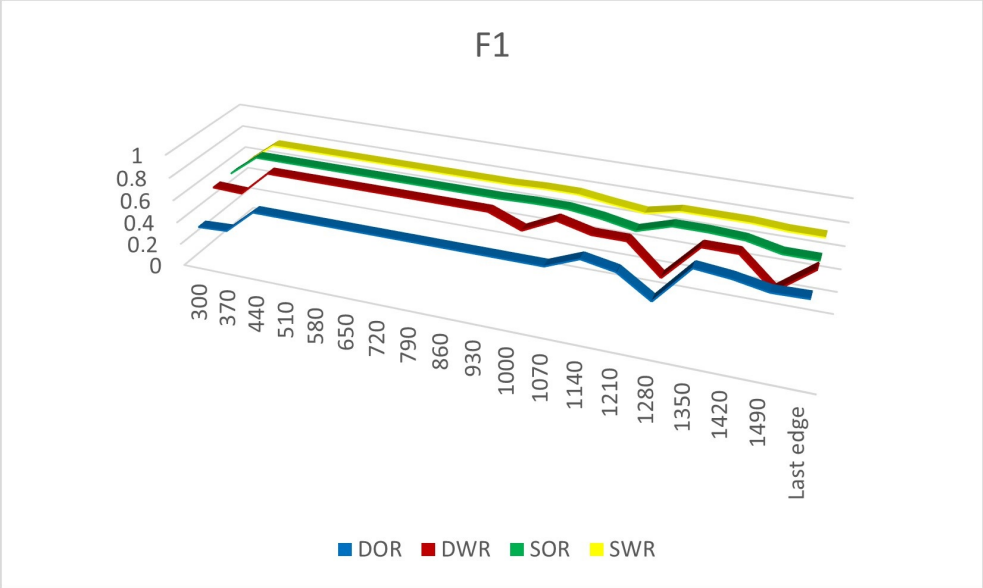
methods work better. Analyzing the dataset, we observe that our anchor (manager) sometimes sends emails in several departments (other communities) and as a consequence, DOR and DWR lose the community coherence while a lot of nodes leave the community. Furthermore, when the batch is reached and the static algorithm is called, the above methods cannot add any new nodes because the fitness score of the already-existing community cannot be increased. For this reason, we have some fluctuations in recall and, as a consequence, in F1 score. On the other hand, the two static algorithms, SOR and SWR, due to the fact that they run from scratch, can add many nodes in C. Thus, the F1 score for the static methods is consistently higher with an average of 91.5%.



	DOR	DWR	SWR	SOR
$b = 30\%$	0.736	0.780	0.915	0.915

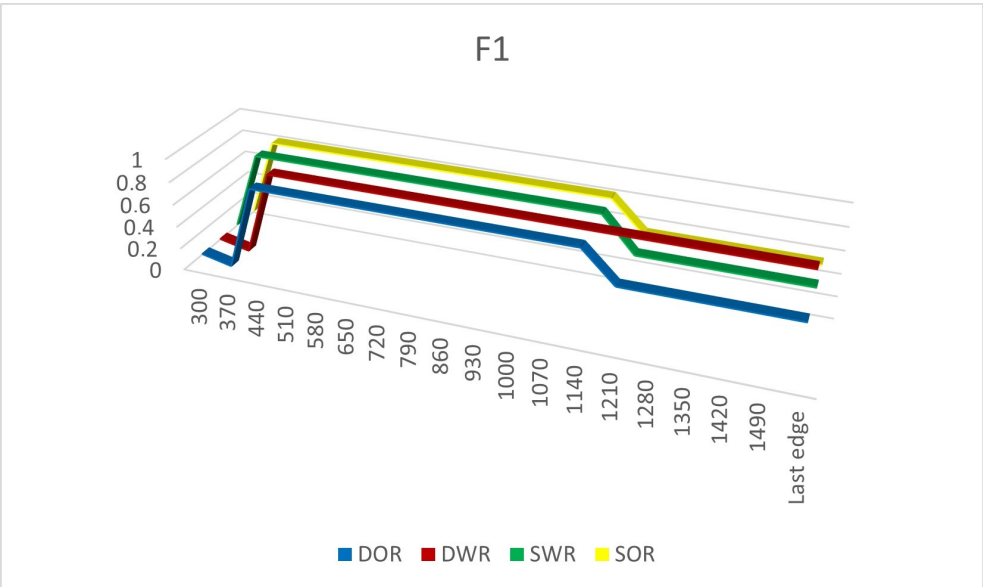
Figure 11: Using a node of high importance from metadata of the email dataset as an anchor. The tuple at the bottom presents the average F1 score for the different methods.

Regarding the experiments in the terrorism dataset, first we present the average of several nodes (terrorists) that have the most important role in a terrorism action (e.g., bomber, foot soldier). In addition, we use a reward equal to 2. In Figure 12 we clearly observe that the proposed method outperforms DOR with almost 14% difference. Furthermore, the results are almost the same when comparing the two static methods. In Figure 13, we used as an anchor a terrorist that had the role of the foot soldier. A foot soldier, e.g., suicide bomber [44], has one of the most important roles in a terrorism organisation. One more time we see that the proposed method produces the best results. More precisely, DWR outperforms the other three methods. Both static methods work better than DOR, and SOR is slightly better than SWR. In this case, we need to point out that, in real networks, the SOR method gives at least the same results with SWR, compared to synthetic graphs where SWR is always better than SOR. The reason is, as mentioned above, that in synthetic datasets there are both insertions and deletions. The rewarding scheme, when a deletion cause the break of the community coherence, helps the static algorithm to add nodes in C and as consequence to improve the recall and the F1 scores.



	DOR	DWR	SWR	SOR
$b = 30\%$	0.582	0.725	0.742	0.743

Figure 12: Average of several nodes of high importance as anchors in the terrorism dataset. The tuple at the bottom present the average F1 score for the different methods.



	DOR	DWR	SWR	SOR
$b = 30\%$	0.657	0.741	0.684	0.696

Figure 13: Using a node of high importance from metadata of the terrorism dataset as an anchor. The tuple at the bottom presents the average F1 score for the different methods.

5 Discussion

The experimental results of Section 4.4 generally show that the LCDS-A framework improves community detection results aggravating slightly the time efficiency. More specifically, several experiments were conducted with different rewards and batch sizes. Our goal was to present the most significant results in terms of the F1 score. More precisely, we focused on the differences in F1 score between

DOR and DWR methods. However, in most of the cases, for both synthetic and real datasets, the proposed framework outperforms even the static algorithms, in terms of the F1 score. On the one hand, these results indicate the importance of giving an extra reward around the anchor's area. On the other hand, it seems that when the community changes slightly between successive updates, it is better to incrementally update the community rather than recomputing it from scratch. Of course, this is not the case when large changes happen at the vicinity of the anchor.

F_{monc} is the metric we used in order to measure the quality of community C and the one that was presented in our experiments. However, for comparison purposes we also used conductance as a quality metric [45]. The results of the experiments with conductance were the same as F_{monc} in the best case. For this reason, only the experiments with F_{monc} are shown in this paper.

Furthermore, different batches and rewards were used. However, the most significant results in F1 score for our datasets were obtained with a batch equal to 30% of the size of the community and a reward equal to 2. Indeed, on the one hand, due to the fact that the static algorithm is triggered less times when the batch size is increased, the resulted communities exhibit low percentages in terms of precision, recall and F1-score. On the other hand, our algorithm is not so time consuming because the static algorithm is called smaller number of times. In addition, in our experimental evaluation, the influence range was set to 1, since otherwise, the process becomes more time consuming combined with lower F1 scores in most of the cases. Better F1 scores were obtained for larger influence ranges in the case when the anchor was in the periphery of a sparse network. In this case, the larger influence range did not have a considerable impact on the time efficiency since the network is sparse and at the same time it identified a non-trivial community around the anchor.

6 Conclusions

Dynamic local community detection is an area of research that has attracted the interest of scientists in recent years. In the present work, we focused on the discovery of local communities that contain important nodes called anchors. Our goal was not only to identify such communities but also to track their evolution over time as new edges are inserted and/or deleted in a network. To achieve this, we proposed a multi-step framework that updates the anchor's community for each incoming edge change in the anchor's influence range or community area. The influence range was used to minimize the avalanche effect. To determine the most stable anchor community, we proposed a node reward method. That is, at each update, we suggested rewarding the edges closer to the anchor by increasing their weight.

An experimental evaluation of the proposed framework was performed on three different synthetic and two real datasets. We used a rather simple but quite efficient reward scheme and we compared the results with the case where no reward method was used. Our results show that our method outperforms the dynamic method without rewards in terms of recall, precision, and F1 score.

From a practical perspective, the proposed approach can help researchers discover useful features of the networks they study. For instance, the suggested approach can be applied to co-purchase networks in order to uncover buying habits. This could lead to smarter advertising techniques and improved targeted marketing. Another example of the practical application of the suggested approach could be politics. For instance, the LCD-A framework could reveal the trends in a network of politics collaborations. Apart from these, another type of network that our proposed approach could be useful for is that of medicine co-prescription. Having patients' demographics and medical history as side information, our framework could uncover the future medical problems of specific anchor patients.

We intend to further extend this work along the following axis: (1) extended experimental evaluation of more rewarding schemes that take into account the edge history, possibly using ageing mechanisms; and (2) extensive tuning of the different parameters of the reward schemes.

References

[1] Fortunato, S. Community detection in graphs. *Phys. Rep.* **2010**, *486*, 75–174.

[2] Veldt, N.; Klymko, C.; Gleich, D.F. Flow-based local graph clustering with better seed set inclusion. In Proceedings of the 2019 SIAM International Conference on Data Mining, Calgary, AL, Canada, 2–4 May 2019; pp. 378–386.

[3] Bian, Y.; Ni, J.; Cheng, W.; Zhang, X. The multi-walker chain and its application in local community detection. *Knowl. Inf. Syst.* **2019**, *60*, 1663–1691.

[4] Bian, Y.; Luo, D.; Yan, Y.; Cheng, W.; Wang, W.; Zhang, X. Memory-based random walk for multi-query local community detection. *Knowl. Inf. Syst.* **2020**, *62*, 2067–2101.

[5] De Meo, P.; Ferrara, E.; Fiumara, G.; Provetti, A. Mixing local and global information for community detection in large networks. *J. Comput. Syst. Sci.* **2014**, *80*, 72–87.

[6] Baltsou, G.; Christopoulos, K.; Tsichlas, K. Local Community Detection: A Survey. *IEEE Access* **2022**, *10*, 110701–110726.

[7] Kostakos, V. Temporal graphs. *Phys. A Stat. Mech. Its Appl.* **2009**, *388*, 1007–1023.

[8] Casteigts, A.; Flocchini, P.; Quattrociocchi, W.; Santoro, N. Time-varying graphs and dynamic networks. In Proceedings of the 10th International Conference, ADHOC-NOW 2011, Paderborn, Germany, July 18-20, 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 346–359.

[9] Baltsou, G.; Tsichlas, K. Dynamic Community Detection with Anchors. In Proceedings of the 10th International Conference on Complex Networks and Their Applications November 30 -December 2, 2021 Madrid, Spain, 2021 . pp. 64-67. Published by the International Conference on Complex Networks and Their Applications

[10] Yang, L.; Shami, A. IoT data analytics in dynamic environments: From an automated machine learning perspective. *Eng. Appl. Artif. Intell.* **2022**, *116*, 105366.

[11] Bonifazi, G.; Cauteruccio, F.; Corradini, E.; Marchetti, M.; Terracina, G.; Ursino, D.; Virgili, L. A framework for investigating the dynamics of user and community sentiments in a social platform. *Data & Knowl. Eng.* **2023**, *146*, 102183. `https://doi.org/https://doi.org/10.1016/j.datak.2023.102183`.

[12] Bonifazi, G.; Cecchini, S.; Corradini, E.; Giuliani, L.; Ursino, D.; Virgili, L. Investigating community evolutions in TikTok dangerous and non-dangerous challenges. *J. Inf. Sci.* **2022**, *2022*, 01655515221116519. `https://doi.org/10.1177/01655515221116519`.

[13] Jalabneh, R.; Syed, H.Z.; Pillai, S.; Apu, E.H.; Hussein, M.R.; Kabir, R.; Arafat, S.Y.; Majumder, M.A.A.; Saxena, S.K. Use of mobile phone apps for contact tracing to control the COVID-19 pandemic: A literature review. *Appl. Artif. Intell. COVID-19* **2021**, *2021*, 389–404.

[14] Radanliev, P.; De Roure, D.; Ani, U.; Carvalho, G. The ethics of shared COVID-19 risks: an epistemological framework for ethical health technology assessment of risk in vaccine supply chain infrastructures. *Health Technol.* **2021**, *11*, 1083–1091.

[15] Radanliev, P.; De Roure, D. Epistemological and bibliometric analysis of ethics and shared responsibility—Health policy and IoT systems. *Sustainability* **2021**, *13*, 8355.

[16] Aggarwal, C.C.; Yu, P.S. Online analysis of community evolution in data streams. In Proceedings of the 2005 SIAM International Conference on Data Mining, Beach, CA, USA, 21–23 April 2005; pp. 56–67.

[17] Duan, D.; Li, Y.; Jin, Y.; Lu, Z. Community mining on dynamic weighted directed graphs. In Proceedings of the 1st ACM International Workshop on Complex Networks Meet Information & Knowledge Management, Hong Kong, China, 2–6 November 2009; pp. 11–18.

[18] Takaffoli, M.; Rabbany, R.; Zaïane, O.R. Incremental local community identification in dynamic social networks. In Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013), Istanbul, Turkey, 10–13 November 2013; pp. 90–94.

[19] Chen, J.; Zaiane, O.R.; Goebel, R. Detecting communities in large networks by iterative local expansion. In Proceedings of the 2009 International Conference on Computational Aspects of Social Networks, Fontainebleau, France, 24–27 June 2009; pp. 105–112.

[20] Yun, S.Y.; Proutiere, A. Streaming, memory limited algorithms for community detection. *Adv. Neural Inf. Process. Syst.* **2014**, *27, pp. 3167 - 3175* .

[21] Zakrzewska, A.; Bader, D.A. A dynamic algorithm for local community detection in graphs. In Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Paris, France, 25–28 August 2015; pp. 559–564.

[22] Zakrzewska, A.; Bader, D.A. Tracking local communities in streaming graphs with a dynamic algorithm. *Soc. Netw. Anal. Min.* **2016**, *6*, 1–16.

[23] Kanezashi, H.; Suzumura, T. An incremental local-first community detection method for dynamic graphs. In Proceedings of the 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, USA, 5–8 December 2016; pp. 3318–3325.

[24] Coscia, M.; Rossetti, G.; Giannotti, F.; Pedreschi, D. Demon: A local-first discovery method for overlapping communities. In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Beijing, China, 12–16 August 2012; pp. 615–623.

[25] DiTursi, D.J.; Ghosh, G.; Bogdanov, P. Local community detection in dynamic networks. In Proceedings of the 2017 IEEE International Conference on Data Mining (ICDM), New Orleans, LA, USA, 18–21 November 2017; pp. 847–852.

[26] Guo, K.; He, L.; Huang, J.; Chen, Y.; Lin, B. A Local Dynamic Community Detection Algorithm Based on Node Contribution. In Proceedings of the CCF Conference on Computer Supported Cooperative Work and Social Computing, Kunming, China, 16–18 August 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 363–376.

[27] Conde-Cespedes, P.; Ngonmang, B.; Viennet, E. An efficient method for mining the maximal α-quasi-clique-community of a given node in complex networks. *Soc. Netw. Anal. Min.* **2018**, *8*, 20.

[28] Liu, J.; Shao, Y.; Su, S. Multiple local community detection via high-quality seed identification over both static and dynamic networks. *Data Sci. Eng.* **2021**, *6*, 249–264.

[29] Liakos, P.; Papakonstantinopoulou, K.; Ntoulas, A.; Delis, A. Rapid detection of local communities in graph streams. *IEEE Trans. Knowl. Data Eng.* **2020**, *34*, 2375–2386.

[30] Rossetti, G.; Cazabet, R. Community discovery in dynamic networks: a survey. *ACM Comput. Surv. (CSUR)* **2018**, *51*, 1–37.

[31] Havemann, F.; Heinz, M.; Struck, A.; Gläser, J. Identification of overlapping communities and their hierarchy by locally calculating community-changing resolution levels. *J. Stat. Mech. Theory Exp.* **2011**, *2011*, P01023.

[32] Rossetti, G. RDYN:graph benchmark handling community dynamics. *J. Complex Netw.* **2017**, *5*, 893–912.

[33] Collection, Stanford Large Network Dataset Available online:`http://snap.stanford.edu/data` accessed on 25 November 2022.

[34] Database, J.J.A.T.T. Al Qaeda Operations Attack Series 1993–2003, Worldwide. 2003. Available online: `http://doitapps.jjay.cuny.edu/jjatt/data.php` accessed on 5 November 2022.

[35] F1 Score Lemma. F1 Score Lemma—Wikipedia, The Free Encyclopedia, Available online: `https://en.wikipedia.org/wiki/F-score` accessed on 5 May 2022.

[36] Li, Y.; He, K.; Bindel, D.; Hopcroft, J.E. Uncovering the small community structure in large networks: A local spectral approach. In Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, 18–22 May 2015; pp. 658–668.

[37] Li, Y.; He, K.; Kloster, K.; Bindel, D.; Hopcroft, J. Local spectral clustering for overlapping community detection. *ACM Trans. Knowl. Discov. Data (TKDD)* **2018**, *12*, 1–27.

[38] Shang, R.; Zhang, W.; Zhang, J.; Feng, J.; Jiao, L. Local community detection based on higher-order structure and edge information. *Phys. A Stat. Mech. Its Appl.* **2022**, *587*, 126513.

[39] Rossetti, G.; Pappalardo, L.; Rinzivillo, S. A novel approach to evaluate community detection algorithms on ground truth. In Proceedings of the Complex Networks VII: Proceedings of the 7th Workshop on Complex Networks CompleNet 2016, Dijon, France, 23–25 March 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 133–144.

[40] Zhang, Y.; Wu, B.; Liu, Y.; Lv, J. Local community detection based on network motifs. *Tsinghua Sci. Technol.* **2019**, *24*, 716–727.

[41] Jaccard Similarity Coefficient Lemma. Jaccard Similarity Coefficient Lemma—Wikipedia, The Free Encyclopedia. Available online: `https://en.wikipedia.org/wiki/Jaccard_index`, accessed on 2 November 2022.

[42] Labatut, V.; Cherifi, H. Accuracy measures for the comparison of classifiers. *arXiv* **2012**, arXiv:1207.3790.

[43] Bharali, A. An Analysis of Email-Eu-Core Network. *Int. J. Sci. Res. Math. Stat. Sci.* **2018**, *5*, 100–104. `https://doi.org/10.26438/ijsrmss/v5i4.100104`.

[44] Gill, P.; Young, J.K. Comparing role-specific terrorist profiles. 2011. SSRN 1782008. Available online: `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1782008`, accessed on 8 November 2022.

[45] Gao, Y.; Zhang, H.; Zhang, Y. Overlapping community detection based on conductance optimization in large-scale networks. *Phys. A Stat. Mech. Its Appl.* **2019**, *522*, 69–79.