CoDÆN: Benchmarks and Comparison of Evolutionary Community Detection Algorithms for Dynamic Networks

GIORDANO PAOLETTI, Politecnico di Torino, Torino, Italy
LUCA GIOACCHINI, Politecnico di Torino, Torino, Italy
MARCO MELLIA, Politecnico di Torino, Torino, Italy
LUCA VASSIO, Politecnico di Torino, Torino, Italy
JUSSARA ALMEIDA, Department of Computer Science, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil

Web data are often modelled as complex networks in which entities interact and form communities. Nevertheless, web data evolves over time, and network communities change alongside it. This makes Community Detection (CD) in dynamic graphs a relevant problem, calling for *evolutionary* CD algorithms. The choice and evaluation of such algorithm performance is challenging because of the lack of a comprehensive set of benchmarks and specific metrics. To address these challenges, we propose CoDÆN– COmmunity Detection Algorithms in Evolving Networks – a benchmarking framework for evolutionary CD algorithms in dynamic networks, that we offer as open source to the community. CoDÆN allows us to generate synthetic community-structured graphs with known ground truth and design evolving scenarios combining nine basic graph transformations that modify edges, nodes, and communities. We propose three complementary metrics (i.e. Correctness, Delay, and Stability) to compare evolutionary CD algorithms.

Armed with CoDÆN, we consider three evolutionary modularity-based CD approaches, dissecting their performance to gauge the trade-off between the stability of the communities and their correctness. Next, we compare the algorithms in real Web-oriented datasets, confirming such a trade-off. Our findings reveal that algorithms that introduce memory in the graph maximise stability but add delay when abrupt changes occur. Conversely, algorithms that introduce memory by initialising the CD algorithms with the previous solution fail to identify the split and birth of new communities. These observations underscore the value of CoDÆN in facilitating the study and comparison of alternative evolutionary community detection algorithms.

CCS Concepts: • **Theory of computation → Dynamic graph algorithms**; • **Information systems** → *Social networks*; • **Computing methodologies** → Simulation evaluation.

Additional Key Words and Phrases: Community Detection, Dynamic Networks, Online Networks, Modularity, Benchmark

1 Introduction

The study of graphs and networks is highly relevant in the context of the Web, due to the inherent structure and complexity of Web-related data [37, 43, 44, 65]. Far beyond being a mere network of documents, the Web serves as a vital platform for social interactions. Social network analysis, which is often focused on dynamic complex

This submission is an extended version of the earlier paper *Benchmarking Evolutionary Community Detection Algorithms in Dynamic Networks* [60].

Authors' Contact Information: Giordano Paoletti, Politecnico di Torino, Torino, Piemonte, Italy; e-mail: giordano.paoletti@polito.it; Luca Gioacchini, Politecnico di Torino, Torino, Piemonte, Italy; e-mail: luca.gioacchini@polito.it; Marco Mellia, Politecnico di Torino, Torino, Piemonte, Italy; e-mail: marco.mellia@polito.it; Luca Vassio, Politecnico di Torino, Torino, Piemonte, Italy; e-mail: luca.vassio@polito.it; Jussara Almeida, Department of Computer Science, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brazil; e-mail: jussara@dcc.ufmg.br.

ACM 1559-114X/2025/2-ART
https://doi.org/10.1145/3718988

networks [19, 77], helps in studying patterns of relationships and influence among users, which is crucial for understanding how online communities and social dynamics evolve over time [33].

Detecting communities, i.e. groups of nodes that exhibit stronger internal connections compared to their connections with the broader network [28], and understanding their temporal dynamics allows a deeper comprehension of the underlying structure and function of the systems they represent [17, 63, 78]. In practical systems, any network topology is expected to change over time, due to the natural growth of the Web in terms of entities (users, content items, etc.), the highly dynamic nature of their connections, and their noisy nature. Such changes may or may not imply a natural evolution of the underlying communities which could evolve at a slower pace than the graph. Finding effective solutions to address CD problems in evolving networks is still an arduous challenge [19, 66] and calls for *evolutionary* CD solutions.

The goal of evolutionary CD techniques [7, 23] is to identify high-quality communities on each snapshot of the dynamic graph while maintaining temporal coherence in the community assignment across successive snapshots. Nevertheless, there is no clear metric to assess and compare the results of evolutionary CD algorithms. As such, it calls for a deep study of the (relative) performance of evolutionary CD algorithms under different and *controllable* scenarios, in which the existence of ground truth allows for a clear understanding of the performance trade-offs.

In this paper, we address these challenges by proposing CoDÆN, a framework for evaluating and benchmarking evolutionary CD algorithms in dynamic networks with known ground truth. Given an initial graph, we design 9 *in-vitro* controllable transformations affecting both edges and nodes, reflecting evolutionary patterns that are intuitively found on the Web. We define three metrics (namely, *Stability*, *Correctness* and *Delay*) that capture complementary and yet desirable properties of evolutionary CD algorithms. *Stability* quantifies the changes in communities over time, whereas *Correctness* and *Delay* are supervised metrics leveraging the generated ground truth to measure how good communities are, and how long the algorithm takes to detect a sudden change. To the best of our knowledge, we are the first to propose a systematic use of the combination of such metrics specifically tailored for this goal. CoDÆN allows us to understand which algorithm performs better in which scenarios. Compared to previous works [15, 69], CoDÆN allows us to generate controlled benchmarks where both edges and (new) nodes evolve over time, and present well-defined and specifically crafted metrics to thoroughly compare algorithms in evolving scenarios.

Although CoDÆN was not designed with a specific family of evolutionary CD algorithms in mind, we validate it here using modularity-based methods, given their widespread application across various domains ranging from social network analysis, to biological systems and bibliometric networks [18, 29, 34, 80]. These are based on the well-known static CD Louvain, or Greedy Modularity Algorithm (GMA) [14], and Leiden algorithms [73]. Modularity is a statistical measure of the quality of a partition of a network into non-overlapping communities that relies on the difference between the actual edges within communities and the expected edges in a random network with the same degree distribution [54]. On a static graph, GMA iteratively merges communities to maximize this measure. Similarly, Leiden improves the greedy search for the best solution by incorporating refinement steps. The two algorithms have been adapted to dynamic scenarios by incorporating memory into the graph or by inheriting initialization from the previous snapshot (see Sections 4.2 and 4.3). Here, we propose a third method that refines the initialization of new nodes and estimates the consistency of the communities identified in the previous step (see Section 4.4). We compare these methods on CoDÆN first, followed by validation on real-world datasets from two distinct Web-related domains, namely network traffic analysis and social network analysis, which exhibit very different network dynamics.

Our results show the trade-off between stability and correctness when evolving communities are present. In a nutshell, the best-suited algorithm depends on the actual transformation the graph presents. CoDÆN is instrumental to both understanding and precisely evaluating such patterns. For this, we offer CoDÆN and the modularity-based evolutionary CD detection algorithms as open source to stimulate the design and evaluation of other evolutionary community detection algorithms, including methods that are not based on modularity, such as *Facetnet* [48], *PisCES* [49] and *DynGEM* [30].[1]

This paper significantly extends our preliminary work introduced in [60], where the evaluation was limited to GMA-based evolutionary CD algorithms. Here, we expand the analysis to include the Leiden algorithm. Additionally, we complement our prior evaluation with a detailed sensitivity analysis of key parameters allowing us to assess the influence of these parameters on algorithm performance. We also conduct an extensive set of experiments to evaluate scalability, gauging the efficiency of each approach on larger datasets. Finally, we validate these methods on real-world datasets where ground truth communities are unknown, providing insights into algorithm performance in unsupervised, real-world scenarios.

The remainder of this paper is organised as follows. In Section 2 we review prior work, and we provide a comprehensive overview of the CoDÆN framework in Section 3. Next, in Section 4 we describe GMA and Leiden as the reference CD algorithms, along with their three evolutionary extensions. We report and discuss our results using both CoDÆN and real Web-oriented datasets in Section 5 and Section 6, respectively. Finally, in Section 7 we draw our final considerations.

2 Related Work

2.1 Community detection for static graphs

Scientific literature is rich in CD approaches for static graphs [39]. Traditional methods rely on the Label Propagation Algorithm [25, 46], which propagates a starting set of labels (i.e. community identifiers) through a network by iteratively updating node labels through majority voting among its neighbours [62]. A second family is based on the Clique Percolation Method [9, 10], which finds cliques, (i.e. fully connected sub-graphs) and links them based on shared nodes [58]. A third family applies spectral clustering algorithms to the graph Laplacian [20, 35, 55]. Such approaches are affected by scalability issues in large networks. Additionally, they require prior knowledge of the number of communities to detect, which is rarely available in real-world scenarios [38]. Besides these methods, Infomap [67] optimizes a *ensamble* of random walks through the network, clustering nodes that are frequently visited together, and it offers an alternative which emphasizes the dynamism of information flow over connectivity. With the spread of Deep Learning technologies, recent works started exploring CD solutions based on Artificial Neural Networks to embed node attributes and topology information. Hence, traditional clustering algorithms are applied on top of the generated representation [4, 22, 51].

Despite the wide literature on CD approaches, modularity-based algorithms, such as Louvain or Greedy Modularity Algorithm (GMA) [14] and Leiden [73] are among the most used in real-world applications. Being scalable and almost parameter-free algorithms, they allow us to find a hierarchical partition of the graph nodes through a greedy optimisation of the modularity, a measure that quantifies the degree to which a network can be divided into non-overlapping and densely connected communities. While we acknowledge that they may not always deliver the best results, their prevalence in the literature and practical effectiveness in numerous applications justify their selection as the focus of our study.

2.2 Evolutionary community detection

Finding effective solutions to address CD problems in evolving graphs is still an arduous challenge [19, 66]. A naïve approach involves independently running a static CD algorithm on successive snapshots of the dynamic graph [15]. This would locally optimize the objective of the static algorithm used in each snapshot (such as the modularity in GMA and Leiden). Yet, it requires an additional step to align communities across time, typically by comparing membership overlap [72], with no guarantee of any temporal coherence. Thus, there is a need

[1]The CoDÆN code, along with the algorithms to reproduce the experiments described in this paper, is available at https://github.com/SmartData-Polito/Dynamic_CommunityDetection_Benchmark

for *evolutionary* CD algorithms, which explicitly consider the evolution of the graph over time while finding communities.

Specifically for modularity-based algorithms, the two main evolutionary extensions have been specifically designed on GMA. They consist of (i) incorporating a memory term into the weights of the graph edges before the application of the standard CD approach (e.g. αGMA) [23, 31]; and (ii) introducing a memory component into the algorithm by initialising nodes as members of their previously assigned communities (e.g. sGMA) [7]. In this work, we consider both approaches and propose a generalization of the sGMA approach that explicitly considers the arrival of new nodes (previously not well-addressed) as well as the local assessment of the consistency of the communities identified in the previous step. We also extend all of them to the Leiden algorithm.

Although modularity-based methods are popular, other techniques offer distinct advantages and may perform better in certain contexts. Some methods, for instance, maintain stability by recalculating only the altered network segments, leaving the unchanged parts intact. This approach underpins algorithms like *Facetnet* [48], which leverages nonnegative matrix decomposition, *ESPRA* [76], a density-based method that assesses network similarity using both topological features and structural perturbations, and *PisCES* [49], which employs evolutionary spectral clustering with feature vector smoothing. Other models use node or subgraph embeddings to capture the dynamic properties of graphs, clustering these representations to reveal community structures. Examples include architectures like autoencoders (e.g., *DynGEM* [30]), Graph Neural Networks (*ROLAND* [81]), and Graph Convolutional Networks (*CTGCN* [52]).

As mentioned, we here choose to test our CoDÆN benchmark on modularity-based methods, due to their widespread adoption in various scenarios. Yet, CoDÆN was not designed specifically for that family of algorithms and indeed can be used to support comparative evaluation across various approaches.

Recently, researchers have directed their efforts toward comparing evolutionary CD algorithms by introducing comparative frameworks [15, 69]. However, their focus remains confined to networks in which the evolution of the graph is based solely on changes on the edges, overlooking scenarios where nodes arrive or vanish from a community. These scenarios are typical in Web contexts (e.g. user inactivity within an email network, users joining a social network, or the appearance of posts in a retweet network). Furthermore, previous comparisons were based on standard metrics that reflect graph properties (e.g. cut ratio and node degree), but lack a comprehensive suite of indicators tailored specifically to evaluate the algorithm performance (the stability of the found solution, the delay in the detection of a change, etc.). To address these concerns, CoDÆN handles the evolution of nodes (including their disappearance, sporadic presence, or addition) and provides a comprehensive evaluation of algorithm performance using tailored metrics such as *Stability* and change detection *Delay* (see Section 3.3).

2.3 Community detection for Web analysis

Since the Web is a large complex network of connected entities (e.g. documents, users, Web pages), it is straightforward to represent Web-related data as graphs. Consequently, the adoption of CD algorithms to analyse communities and identify their patterns is a known practice [13, 32, 70]. Common applications are the enhancement of end-users Quality of Experience, as in the identification of Web pages hosting similar contents for a faster indexing [56] and the identification of communities exhibiting similar interests to fasten edge caching in online services [3, 68], or the detection of malicious communities in e-commerce platforms preventing online frauds [16, 47, 50].

Among the many diverse applications, CD techniques are particularly suitable for analysing Web-related data, with a special focus on social network analysis [12, 33]. Social media are often modelled as dynamic complex networks, or graphs, where entities (nodes) interact with each other establishing connections (edges) that evolve over time [19, 77], often exhibiting strong community structures. Analysing such communities allows to understand some real-world events like the spread of fake news [21, 53], the vaccine debate during the COVID-19 pandemic [59] and political polarisation [11], like the 2020 U.S. election fraud debate [2].

The growing concern about security and privacy issues within the Web has pushed researchers to employ CD algorithms in cybersecurity applications. Notable examples rely on building a similarity network of packages used by Android mobile apps to fingerprint families of malware [6, 41, 42]. Successful results have been obtained in traffic analysis applications, where modelling network traffic as a graph allows to identification of communities of hosts exhibiting similar behaviours within a network and to detection of emerging threats (e.g. botnets) [26, 27, 40].

In this work, we consider two real-world datasets that are examples of evolving graphs modeling different phenomena of the Web: the 2020 U.S. election fraud debate on Twitter [2] and new attacks by botnets or Internet scanners [27].

3 CoDÆN: Framework Overview

COmmunity Detection Algorithms in Evolving Networks, or CoDÆN, consists of a combination of (i) synthetically generated dynamic graphs with known community structures; (ii) a set of transformations involving temporal changes in both node and edge sets that cover different realistic scenarios; and (iii) three complementary metrics that enable the evaluation of the performance of alternative evolutionary CD algorithms.

We define a dynamic graph $\{\mathcal{G}^t\}_{t=1}^{N}$ as a sequence of N undirected graphs $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t)$, where $\mathcal{V}^t$ and $\mathcal{E}^t$ are, respectively, the set of nodes and edges at snapshot t.[2] Namely, an edge $\epsilon = (u, v, w^t) \in \mathcal{E}^t$ indicates a connection between nodes $u, v \in \mathcal{V}^t$ with weight $w^t \in (0, 1]$ at snapshot t. Focusing on a single snapshot[3], we define a community $c \subseteq \mathcal{V}$ as a set of nodes with a higher density of internal connections compared to their connections with the rest of the network. Hence, $\bigcup_{c \in C} c = \mathcal{V}$, where C is the set of detected communities. In this work, we focus on non-overlapping communities.

3.1 Generating synthetic graphs

The evaluation of evolutionary CD algorithms is often challenged by the lack of ground truth. Thus, we propose synthetic scenarios reflecting transformations in the graph topology that resemble real-world evolving networks and that offer a known ground truth against which alternative solutions can be compared.

Specifically, we run *in-vitro* experiments generating synthetic graphs through the Lancichinetti-Fortunato-Radicchi (LFR) benchmark [45], a widely recognized tool for creating complex networks exhibiting community structures and providing known ground-truth (i.e. the assigned communities). Although any synthetic community-based graph generation algorithm could be used, we here choose LFR as it is known to generate realistic graphs, characterised by power-law degree distributions and heterogeneous community sizes. We randomly initialize weights to the edges with values uniformly distributed in the $(0, 1]$ range.

3.2 Graph transformations

We propose 9 transformations, grouped into 3 scenarios – i.e. (i) Noise, (ii) Morphing and (iii) Disruptive. These transformations can be applied at different rates. We provide a concise overview of the designed scenarios in Table 1. Despite simple, these transformations decompose the complex dynamics of real-world graphs into individual events, allowing the identification of algorithmic weaknesses, and they are inspired by previous work [15, 57, 66]. As future work, our framework could be extended to explore combinations of them to create more intricate transformation sequences.

[2]Each snapshot can represent a single edge or node change in a continuous-time manner or can group changes in batches in a periodic manner.

[3]We omit the suffix t for the sake of clarity.

Table 1. Overview of the CoDÆN basic graph transformations.

Scenario	Transformation	Description
Noise	Expansion	Add nodes within communities
	Intermittence	Turn off/on nodes for *one* snapshot
	Switch	Turn off/on nodes for *multiple* snapshots
Morphing	Merge	Merge community pairs
	Split	Form sub-communities
	Death	Dissolve communities
	Birth	Aggregate nodes in new communities
Disruptive	Mixing	Decay of community structure
	Removal	Remove nodes

3.2.1 Noise scenario. In this scenario, we perturb the graph without affecting the initial ground truth provided by the LFR graph through 3 transformations:
(i) *Expansion.* At each snapshot, we progressively add new nodes to existing communities. We preserve the node degree heterogeneity by randomly selecting nodes to connect to within the community through preferential attachment [8] as in the initial LFR network. The new nodes persist throughout the simulation.
(ii) *Intermittence.* At each snapshot t, we remove a fraction ϕ_{int} of nodes and we re-add them at $t+1$. To ensure the persistence of hub nodes characterising the communities, each node k has probability $p_k \propto \frac{1}{\delta_k}$ of being selected, where δ_k is the node degree.
(iii) *Switch.* At each snapshot t, we remove a fraction ϕ_{swi} of nodes. Each node k is removed with probability $p_k \propto \frac{1}{\delta_k}$. Inactive nodes reappear with probability $p_{back} \propto \exp^{t_{off}-\gamma}$, where t_{off} is the number of snapshots the node has already been inactive, and $\gamma \in \mathbb{N}$ is the parameter that controls the average duration of inactivity.

These transformations reflect real-world scenarios, such as users in a social media platform, where the activity exhibits an on-off pattern and new users may join the system without necessarily changing the underlying community structure.

3.2.2 Morphing. In this scenario we progressively change the community structure and the network topology over time. We start with the initial ground truth GT^0 for communities and end up with the final ground truth communities GT^N.
i) *Split.* We select n communities and progressively split them into two sub-communities by decreasing at each snapshot (by an amount τ) the weights of edges connecting the sub-communities.
(ii) *Merge.* We select n community pairs and progressively merge them into a larger one by increasing at each snapshot by τ the weights of the edges connecting their nodes.
(iii) *Death.* We select n communities and progressively dissolve them by decreasing at each snapshot by τ the intra-community edge weights; at the same time, we strengthen the edges of such nodes directed to other existing communities through preferential attachment with respect to their degree;
(iv) *Birth.* We decrease (by an amount τ) the edge weights of some nodes per community and establish connections among them forming n new communities.

We control the speed at which the transformations occur through the parameter $\tau \in [0, 1]$, which expresses the variation in edge weights at each snapshot. At snapshot t we obtain the weight of an edge w^t by modifying the weight of the previous snapshot w^{t-1}. Formally, $w^t = w^{t-1} \pm \tau$, and we clip $w^t \in (0, 1]$.

These transformations reflect transitions that may occur during the rise of new communities in social networks centered on a specific topic or, conversely, the waning interest in certain subjects.

3.2.3 Disruptive. In this scenario, we progressively destroy the community structure of the network. We define two transformations:
(i) *Mixing*. At each snapshot, we shuffle the initial communities by reassigning the destination of an edge from a fraction ϕ_{dis} of nodes, while preserving the weighted degree distribution. (ii) *Removal*. At each snapshot, we progressively remove a fraction ϕ_{dis} of nodes selected uniformly at random.

These transformations reflect realistic scenarios marked by a shift towards a disordered arrangement of nodes, like social mixing, when nodes from different communities randomly connect with each other, or when some nodes leave the network, like when users are no longer interested in it.

3.3 Quality metrics

The metrics used in static CD evaluation (e.g., *Conductance* [61, 79] – the percentage of edges that cross the cluster border; *Internal Density* [61] – the ratio of edges within the cluster w.r.t. all possible edges;) are all unsupervised metrics (i.e. they do not require prior knowledge on the discovered communities). Hence, they do not offer insights about the correctness of the detected partitions. Additionally, they reflect specific properties of the graph, without providing information on the performance of the CD algorithm in evolving graphs through the various snapshots. Here, we define three metrics capturing complementary factors: *Stability* (unsupervised), *Correctness* and *Delay* (supervised)[4]. These metrics rely on the Adjusted Mutual Information (AMI) [75], i.e. a similarity measure between label sets adjusted for random chance, normalised to range from 0 to 1.

Given the communities C^t found at the snapshot t, we define the metrics as follows:

Stability (S). Formally, $S^t = \text{AMI}(C^{t-1}, C^t)$. It is a proxy for the temporal consistency in community assignment. A higher stability indicates fewer changes in community membership between two snapshots. We compute also the overall stability S by averaging S^t over the N simulated snapshots.

Correctness (K). Given a ground truth (GT), we evaluate the correctness of the considered evolutionary CD algorithm in detecting the transformations at snapshot t as $K^t = \text{AMI}(\text{GT}, C^t)$. In the noise scenario, where the transformations do not affect the initial ground truth, we set GT=GT^0. Conversely, in the morphing scenario, since we cannot define the final ground truth until the transformation has ended, we consider the correctness K at the final snapshot, setting GT=GT^N. In disruptive transformations, in turn, the final ground truth GT^N is not available and we do not consider the correctness.

Delay (D). It is the number of snapshots the algorithm takes to detect the change to a new community structure. We define this metric only in morphing transformations and use it to assess the *responsiveness* of each algorithm to detect the transformation. Namely, we define the Crossing Point CP as the first snapshot at which the community assignment is closer to GT^N compared to GT^0. Formally, $\text{CP} = \min\{t : \text{AMI}(\text{GT}^N, C^t) > \text{AMI}(\text{GT}^0, C^t)\}$. We consider the delay D as the difference between the crossing point and the snapshot at which the transformation begins. If the crossing point does not exist, i.e. the change was not detected, we set the delay to N.

4 Modularity-based evolutionary CD algorithms

We design CoDÆN as a tool to evaluate whatever evolutionary CD algorithms. Yet, given their wide use in real-world applications, we focus our evaluation on *modularity-based algorithms*.

Given a set of non-overlapping communities C, the graph *modularity* [54] quantifies how well-defined the communities are. In a nutshell, modularity compares the density of the within-community connections to the connections expected in a random network. Formally, for community $c \in C$, its modularity is defined as $Q_c = \frac{w_{in,c}}{w^*} - \left(\frac{w_c}{2w^*}\right)^2$, where w^* is the sum of all edge weights of the graph, $w_{in,c}$ is the sum of the intra-community

[4]Notice that the latter two require information of the ground truth. This calls for synthetic data.

edge weights, and w_c is the sum of the weighted degree of the nodes in community c. The modularity of the graph is defined as the sum of the modularity of its communities, i.e. $Q = \sum_{c \in C} Q_c$.

4.1 Louvain (GMA) and Leiden for static graphs

The Louvain method, known as the Greedy Modularity Algorithm (GMA) [14], operates by greedily and iteratively maximizing the graph modularity. At the first iteration, GMA assigns each node to a unique community. At each subsequent iteration, it merges communities by evaluating the modularity gain resulting from moving a node to a neighbouring community or merging two adjacent communities. If the modularity gain is positive, the change is accepted, and the process continues. We refer to [14] for further details.

The Leiden algorithm [73] improves the GMA by incorporating refinement steps. At each iteration of GMA – where communities are merged to optimize graph modularity – Leiden introduces two key modifications: firstly, it relaxes constraints on modularity increase, allowing a node to merge with any community that exhibits modularity improvement; secondly, it refines the detected partition by selectively splitting communities that contribute to lower the graph modularity.

GMA or Leiden can be naïvely used in evolutionary graphs by applying either algorithm on each snapshot independently [15]. We refer to this approach as *independent GMA* and *independent Leiden*, respectively. This strategy may lead to temporal inconsistencies in the communities due for instance to the random initialization of labels at the first iteration. Researchers have proposed two evolutionary approaches to minimize such inconsistencies by adding memory to the graph itself or by introducing a memory component to the algorithm. Although these approaches were originally proposed and evaluated as extensions of GMA [7, 23], we here also consider them as extensions to Leiden.

4.2 Introducing memory to the graph (αGMA and αLei)

This approach [23] consists of introducing the persistence of edges over time, increasing community stability. The approach introduces a memory term α in the edge weights. Namely, a previously existing edge ϵ whose weight during snapshot t is $\hat{w}^t$ and was $w^{t-1} \neq 0$ in $t-1$ has its actual weight updated to

$$w^t = (1-\alpha) \cdot \hat{w}^t + \alpha \cdot w^{t-1}.$$

If $w^{t-1} = 0$, we set $w^t = \hat{w}^t$. The application of independent GMA or Leiden on such a graph results in gradual modularity adjustments as new information becomes available. *We refer to these methods as αGMA and αLei.*

4.3 Introducing memory to the algorithm (sGMA and sLei)

This approach changes the initialisation process of GMA and Leiden. While the original method assigns each node to a unique distinct community at snapshot t, this approach maintains the previous community membership for nodes active also in $t-1$. Each new node is assigned to a new distinct community. Given this initialisation, we run the original CD algorithm allowing nodes to adjust their community membership if the initial assignment is not suitable. *We refer to the application of this approach to GMA and Leiden as sGMA and sLei, respectively.*

4.4 Novel Neighbourhood-based Approach (NeGMA and NeLei)

Both sGMA and sLei inherit the community structure from the previous snapshot, while newly appeared nodes are assigned to a new community. To improve this choice, we propose to initialise the label of new nodes relying on the majority class of their neighbourhood in the current snapshot. We next refine the initialization by assessing the consistency of each community, retaining it only if its local modularity exceeds a threshold θ_Q.

For illustration purposes, we provide an overview of how our approach works in Figure 1. Namely, (i) we start from the communities detected at the previous snapshot $t-1$; (ii) at snapshot t, the graph evolves as some new edges (red lines) and nodes (red squares) appear, and some disappear (dashed elements); (iii) we assign already

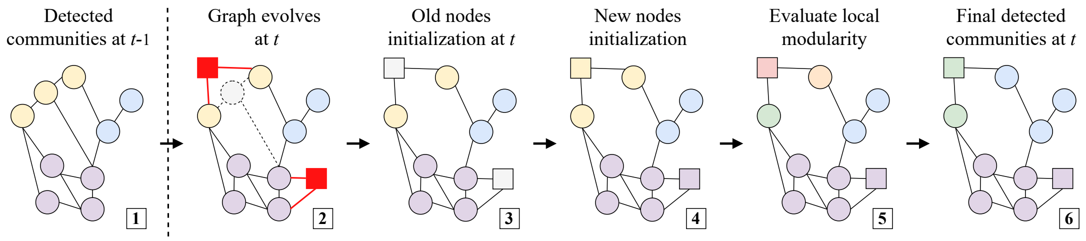

Detected communities at t-1
Graph evolves at t
Old nodes initialization at t
New nodes initialization
Evaluate local modularity
Final detected communities at t
1
2
3
4
5
6

Fig. 1. Neighbourhood-based approach: proposed initialisation of nodes at snapshot t starting from snapshot $t-1$. Different colours represent different communities; squares represent new nodes; dashed elements represent disappearing nodes and edges.

existing nodes (coloured circles) to the communities detected at $t-1$ (as in sGMA/sLei); (iv) we assign new nodes to a community (coloured squares) through majority voting on the labels of their *neighbours*; (v) for each community c, we compute the gain in local modularity with respect to the previous snapshot: $\Delta Q_c^t = Q_c^t - Q_c^{t-1}$. If ΔQ_c^t is lower than a threshold θ_Q, we unbind the community assigning each of its nodes to a distinct community like traditional GMA/Leiden (former yellow community in the upper-left part of the graph at step iv); finally, (vi) we apply independent GMA/Leiden with the final community initialisation. *We refer to the algorithms derived from applying this approach to GMA and Leiden as neighbourhood-based GMA or neighbourhood-based Leiden (or simply NeGMA or NeLei), respectively).*

We present the pseudocodes for the neighbourhood-based approach, for *αGMA/αLei* and *sGMA/sLei* in Appendix A.

5 Evaluation in Synthetic Graphs

We evaluate the evolutionary CD algorithms using CoDÆN benchmark. We repeat each experiment generating 100 independent dynamic graphs. We set the LFR parameter $\mu = 0.2$ [45]. For each dynamic graph, we generate $N = 150$ temporal snapshots. We trigger the beginning of the transformations at snapshot 25 and conclude it at snapshot 125. For each dynamic graph, we perform 10 independent runs of each algorithm on each snapshot. To address the variability introduced by (i) the graph generation phase and (ii) the randomness of the algorithms, we average the results of the 10 runs on each dynamic graph. Consequently, we evaluate the median and bootstrapped 99% confidence interval for these medians across the 100 dynamic graphs.

Regarding the transformations, we set the fraction of nodes for intermittence as $\phi_{int} = 0.2$; for switch $\phi_{swi} = 0.005$ and $\gamma = 10$; for mixing and removal, we uniformly sample ϕ_{dis} between $[0.005, 0.02]$. When not explicitly stated, we set the transformation speed $\tau = 1/100$, and the number of affected communities $n = 3$.

Regarding the algorithms, as suggested in Elgazzar et al. [23], we set $\alpha = 0.8$ in αGMA and αLei, whereas for NeGMA and NeLei we set $\theta_Q = 0$ as a generic intermediate value.

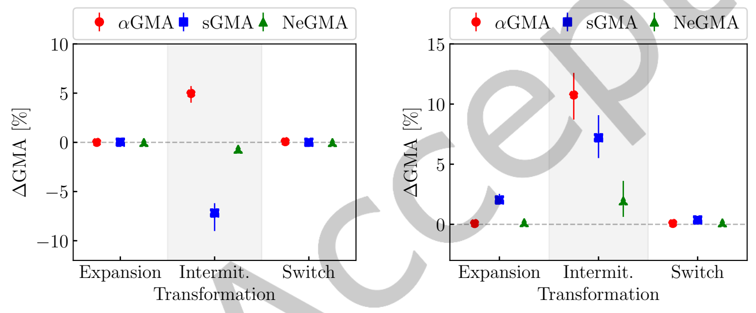
5.1 Correctness and Stability

5.1.1 GMA-based algorithms. In Table 2 we summarise GMA-based algorithms' performance across all simulated scenarios. The signs refer to the relative performance gain/loss obtained by each algorithm over using independent GMA, considered as baseline. No single algorithm excels in all scenarios: each provides a distinct trade-off depending on the transformation. In terms of stability, all evolutionary algorithms perform as well as or better than the baseline. However, this is often obtained at the cost of a decrease in terms of correctness.

Table 2. Overview of Correctness K and Stability S of the tested GMA-based evolutionary algorithms compared to the median of the independent GMA (baseline). '=' indicates a performance variation $\leq$ 5%. A number n of '+' indicates an improvement in the $(\,5n\%\,, 5(n+1)\%\,]$ range. A number n of '−' indicates a decrease in the $[\,-5(n+1)\%\,, -5n\%\,)$ range.

		Noise Scenario			**Morphing Scenario**				**Disruptive Scenario**	
		Expansion	**Interm.**	**Switch**	**Merge**	**Split**	**Birth**	**Death**	**Mixing**	**Removal**
$K^{\dagger}$	GMA	0.84	0.95	0.99	1.00	0.86	0.91	1.00	n.a.	n.a.
	αGMA	=	=	=	=	=	=	=	n.a.	n.a.
	sGMA	=	−	=	=	− − −	− − − −	=	n.a.	n.a.
	NeGMA	=	=	=	=	−	−	=	n.a.	n.a.
S	GMA	0.98	0.90	1.00	0.98	0.93	0.88	0.87	0.99	0.99
	αGMA	=	++	=	=	=	=	=	=	=
	sGMA	=	+	=	=	+	++	++	=	=
	NeGMA	=	=	=	=	=	+	+	=	=

[†] Recall that we compute differently the correctness for noise and morphing scenarios (see Section 3.3).



(a) Correctness (b) Stability

Fig. 2. *Noise and GMA-based algorithms*. Median gain and bootstrapped 99% confidence interval in Correctness and Stability.

Noise transformations In Figure 2 we report *median* correctness and stability gain/loss compared to independent GMA in the noise scenario, spanning from snapshots 25 to 125. All algorithms perform on par with GMA for the expansion and switch transformations, with a gain of up to 2%.

The memory term in αGMA enhances robustness against intermittent noise by boosting both correctness (+5%) and stability (+11%). Indeed, Figure 3b qualitative shows that communities detected by αGMA at snapshot 15 stay consistent up to snapshot 90, unlike independent GMA, which instead fluctuates and fails in maintaining the original setup, as shown in Figure 3a. sGMA initialisation, in turn, improves the stability by 7% for intermittent transformation but loses 7% in correctness compared to independent GMA. In the showcase of Figure 3c, we see that intermittent noise causes sGMA to merge communities at snapshot 60, and the initialization of the algorithm in subsequent snapshots hinders further detection and reconstruction of the original community structure. Conversely, NeGMA overcomes sGMA's limits thanks to the local modularity evaluation, balancing correctness and stability with a correctness decrease $< 1\%$ and stability increase $\approx 3\%$. Supporting this, we see in the example in Figure 3d a partial recovery of the original communities between snapshots 45 and 75.

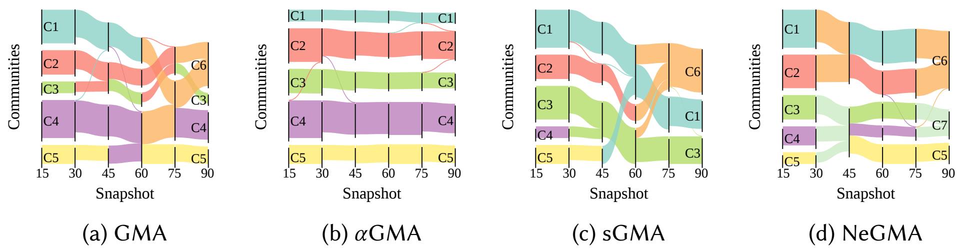


Fig. 3. *GMA-based algorithms*. Example of dynamic evolution of the graph perturbed by the intermittence transformation, showcasing community assignment. Zoom in a sample of snapshots from 15 to 90 with step 15 where changes occur.

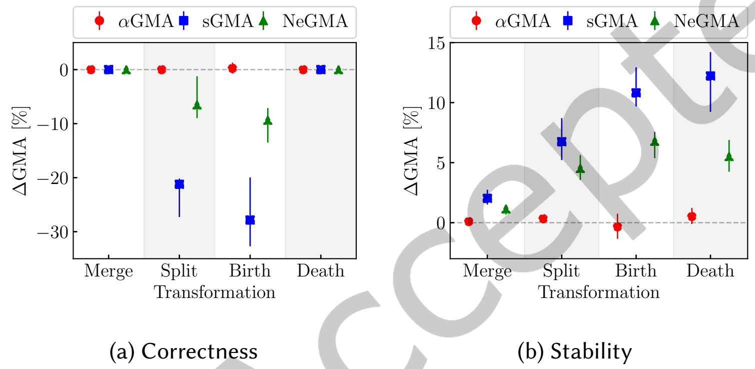
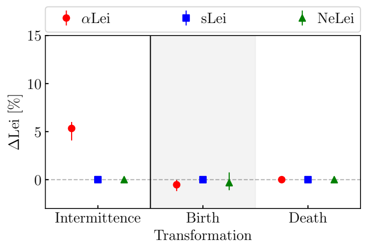


Fig. 4. *Morphing and GMA-based algorithms*. Median gain and bootstrapped 99% confidence interval in Correctness and Stability.

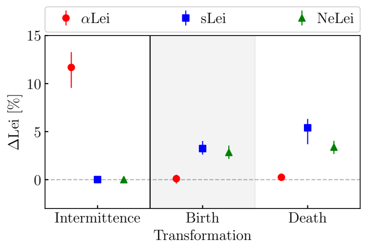
Morphing transformations Figure 4 reports the *median* correctness and stability gain/loss resulting from morphing transformations. Notably, αGMA performs comparably to independent GMA across all designed transformations. In turn, sGMA leads in stability with improvements varying from +3% for merge to +13% for deaths. Yet, it suffers in correctness, with losses above 20% for split and birth transformations, reflecting its difficulty in identifying new communities. NeGMA once again proves to strike the best balance between correctness and stability, with an overall 7% maximum stability gain while limiting correctness to no more than $\approx -10\%$ loss for split and birth transformations.

We also tested the disruptive scenario (Table 2, complete results omitted for the sake of brevity): none of the tested algorithms leads to substantial improvements compared to independent GMA, achieving 0.99 of stability.

5.1.2 Leiden-based Algorithms. In most of the transformations, the community refinement stage introduced by Leiden is extremely effective in detecting the correct communities after the transformation. Hence none of the evolutionary extensions of Leiden brings a substantial improvement compared to its independent execution, except for the following three transformations: intermittence, birth, and death. Figure 5 reports the *median* correctness and stability gain/loss resulting from these three transformations.



(a) Correctness.



(b) Stability.

Fig. 5. *Leiden-based algorithms.* Median gain and bootstrapped 99% confidence interval in Correctness and Stability for the three transformations with larger differences. The vertical solid line distinguishes between noise and morphing scenarios.

The inclusion of a memory term in αLei graph generation process ignores the topology changes introduced in the intermittence transformation. As in αGMA, this enables a notable enhancement in correctness by 5% and stability by 12% compared to independent Leiden (which in fact keeps modifying the communities to reflect the coming and going of nodes).

In the morphing scenarios (birth and death), all algorithms have similar correctness. sLei is confirmed as the most stable algorithm resulting in stability gains of +4% and +7%, respectively. αLei performs comparably to independent Leiden and NeGMA shows slightly less stability than sLei.

Given that Leiden appears insensitive to the employed evolutionary approach in most transformations, throughout the rest of the paper *we focus on comparing the GMA-based evolutionary algorithms.*

Takeaway: (i) Memory terms in αGMA and αLei cause minimal changes compared to independent GMA and Leiden, except in intermittence transformations; (ii) sGMA improves stability but fails in detecting emerging communities, while NeGMA proves balanced in morphing transformations, achieving competitive performance in both correctness and stability; (iii) For most transformations, Leiden is unaffected by evolutionary initializations, except in morphing (birth and death), where both sLei and NeLei improve independent Leiden stability by up to $\approx$ 5% without losing in correctness.

5.2 Responsiveness

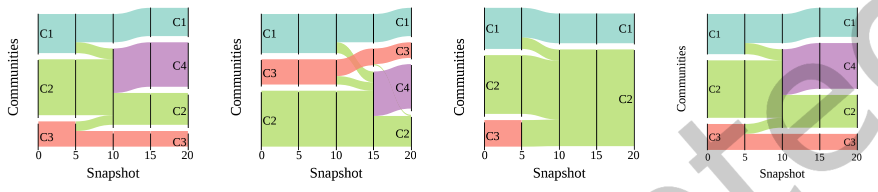
We assess the responsiveness of evolutionary GMA-based solutions in detecting changes by monitoring the delay metric in morphing transformations. We generate 100 independent graphs for each transformation and run 10 instances of each algorithm for $N = 20$ simulated snapshots. We perform an instantaneous (i.e. $\tau = 1$) transformation at snapshot $t = 10$ and measure the delay D to detect it. Given the setup, D falls in the $[0, 10]$ range. Longer delays imply lower responsiveness.

Table 3 reports the median delay for each transformation and the fraction of detected transformations (that is, the crossing point CP is defined[5]). As expected, independent GMA detects the transformation at the exact snapshot with $D = 0$ for all the cases. The memory term of αGMA, in turn, forces the overall graph to evolve at a slower rate, resulting in a systematic delay and lower responsiveness (from $D = 1$ for splits to $D = 4.6$ for births). This is confirmed by the example visualized in Figure 6b, in which the fourth community is detected with a delay of 5 snapshots compared to independent GMA in Figure 6a.

[5]We recall that the Crossing Point (CP) is the first snapshot where the community assignment aligns more closely with the final ground truth GT^N than with the initial ground truth GT^0.

Table 3. *GMA-based algorithms.* Median delay and reached crossing points in *morphing* scenario. Best results are highlighted in **bold**, critical results in **red**.

	Delay D				Reached CP [%]			
	Merge	**Split**	**Birth**	**Death**	**Merge**	**Split**	**Birth**	**Death**
GMA	**0.0**	**0.0**	**0.0**	**0.0**	**0.98**	0.88	0.90	**1.00**
αGMA	3.0	1.0	4.6	2.0	**0.98**	0.90	0.90	0.96
sGMA	**0.0**	**Max**	**0.0**	**0.0**	**0.98**	0.04	**0.98**	0.82
NeGMA	**0.0**	**0.0**	**0.0**	**0.0**	**0.98**	**0.94**	0.88	**1.00**



(a) GMA (b) αGMA (c) sGMA (d) NeGMA

Fig. 6. Example of the dynamic evolution of the graph with instantaneous birth transformation at time 10, showing the community assignment (GMA-based algorithms).

sGMA matches the delay of independent GMA for merges, births, and deaths but misses 96% of the CPs in splits. By preserving the previous snapshot assignment, sGMA fails to detect emerging communities. This limitation also affects birth transformations: while sGMA detects 98% of CPs, it remains tied to the GT^0 assignment (as shown in Figure 6c), thus failing to capture new communities and resulting in a −37.8% correctness drop-off on GT^N.

Finally, NeGMA detects instantaneous transitions with a median delay $D = 0$ for all the cases. Its evaluation of the local modularity ΔQ_c^t during node initialisation addresses the sGMA limitation, refining the initial assignment and successfully detecting 94% of splits. Notably, even though it reaches 10% fewer CPs than sGMA, NeGMA successfully detects the transformations, as highlighted in Figure 6d, resulting in a correctness of 0.97, compared to the 0.60 of sGMA on GT^N.

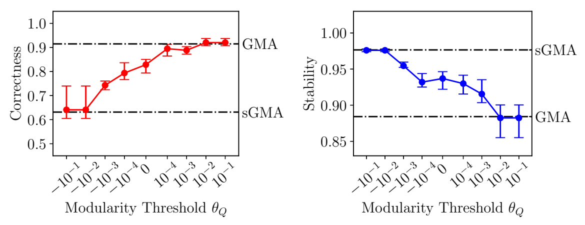
Takeaway: (i) The memory term of αGMA introduces a systematic delay in detecting instantaneous transformations; (ii) sGMA is unable to capture the sudden emergence of new communities; while (iii) NeGMA exhibits high responsiveness and detection rates in most of the instantaneous morphing transformations.

5.3 Sensitivity Analysis

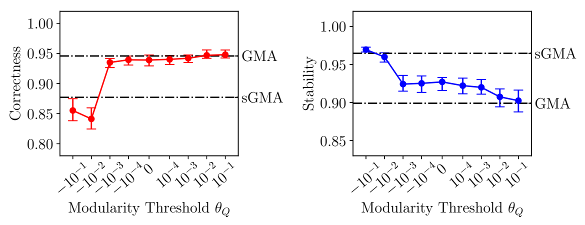
In this section, we investigate the effects of key algorithm parameters in terms of correctness and stability. Namely, we assess the impact of the modularity threshold θ_Q for NeGMA and the memory term α for αGMA.

5.3.1 NeGMA: Impact of modularity threshold θ_Q. We rely on the same 100 synthetic graphs generated through CoDÆN as in Section 5.1. For each graph, we systematically vary the parameter $\theta_Q \in [-0.1,\ 0.1]$. Then, we perform 10 runs of NeGMA for each value, averaging the results.

Recall that parameter θ_Q serves as a gauge for the maximum tolerable modularity decrease within a community following the first NeGMA initialisation. A higher positive threshold makes NeGMA less responsive to such decreases, aligning the initialisation of existing nodes with sGMA and maintaining the communities from the



(a) Birth.



(b) Intermittence.

Fig. 7. Impact of modularity threshold θ_Q on correctness and stability of NeGMA for birth (morphing) and intermittence (noise) transformations.

prior snapshot. On the contrary, a more negative θ_Q amplifies sensitivity to modularity drops within communities, implying the re-initialisation of all the existing nodes. In this case, NeGMA behaves as an independent GMA.

These observations are confirmed in Figure 7, which reports the impact of θ_Q on correctness and stability results for the birth (morphing) and intermittence (noise) transformations.[6] For the sake of comparison, results for independent GMA and sGMA are also shown (horizontal dashed lines).

Let us first focus on the birth transformation with results shown in Figure 7a. Here, when $\theta_Q \leq -0.1$, NeGMA coincides with sGMA achieving ≈ 0.60 of correctness and ≈ 0.98 of stability. Conversely, with $\theta_Q \geq 0.01$, NeGMA becomes less sensitive to modularity declines, performing akin to independent GMA (≈ 0.90 of correctness and ≈ 0.89 of stability). When θ_Q falls withing the range $[-0.1,\ 0.01]$, correctness progressively improves by 30% corresponding to a stability decrease by 9%. Similar considerations emerge for the noise scenario, showcased in Figure 7b. Here, the impact of θ_Q on algorithm stability aligns with the trends seen in the birth transformation. However, concerning correctness, $\theta_Q \leq -0.01$ induces a marginal performance dip of $\approx 2\%$ compared to sGMA. Notably, sharper performance shifts manifest for $\theta_Q > -0.01$.

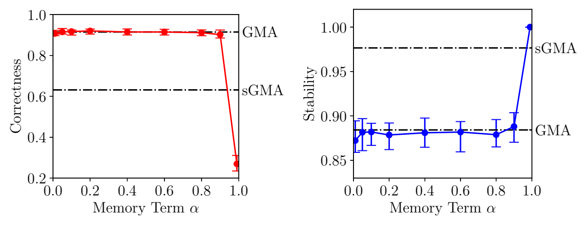
All in all, these findings reaffirm NeGMA as a generalised modularity-based algorithm. Fine-tuning the θ_Q parameter allows for striking a balance between stability and correctness, tailoring performance to the specific graph evolution under consideration.

5.3.2 αGMA: Impact of memory term α. The memory term α on αGMA weights the importance of historical information against more recent data when updating the graph. For $\alpha = 0$ edge weights are set considering only the weights at the current snapshot (as in independent GMA), whereas for $\alpha = 1$ edge weights are set as the ones of the previous snapshot. We rely on the same 100 graphs generated through CoDÆN and vary $\alpha \in [0.05, 0.99]$. For each value and each graph, we run 10 instances of αGMA algorithm, averaging the obtained results.

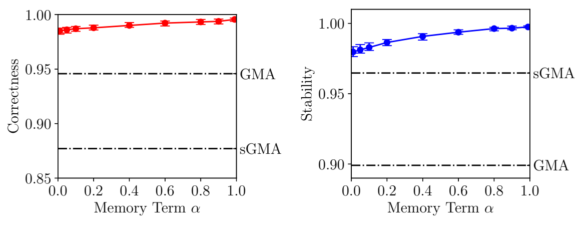
Figure 8 shows the impact of α on correctness and stability for birth and intermittence transformations, as examples of morphing and noise scenarios, respectively[7]. Firstly, for morphing transformations (Figure 8a) the impact of α in negligible. For $\alpha \leq 0.90$, αGMA aligns with independent GMA in terms of both stability and correctness. Conversely, for $\alpha = 0.99$, the contribution of the edge weights at the current snapshot is marginal (i.e. 0.01), hence the algorithm prevents the graph from evolving, preserving the past structure. As a consequence, correctness drastically decreases from 0.9 to 0.3, whereas stability improves from 0.88 to 1.

When focusing on the intermittence transformation in Figure 8b, results are consistent with the considerations of Section 5. Notably, αGMA exhibits improvements in both correctness and stability compared to independent GMA and sGMA. Because of the sporadic presence of nodes due to the considered transformation, even a small

[6]We evaluated all 9 transformations achieving similar qualitative results.
[7]We tested all 9 transformations, achieving similar results.



(a) Birth.



(b) Intermittence.

Fig. 8. Impact of memory term α on correctness and stability of αGMA for birth (morphing) and intermittence (noise) transformations.

value of $\alpha = 0.05$ allows to smooth the changes in the graph topology preserving past connections between nodes. This leads to $\approx 4\%$ of gain in correctness (0.99 of αGMA compared to ≈ 0.94 of independent GMA) and $\approx 2\%$ of stability gain (0.98 of αGMA compared to ≈ 0.96 of sGMA). A value of $\alpha = 0.99$ results in an almost static graph, achieving 100% of both correctness and stability.

Takeaway: (i) Lower values of the threshold θ_Q make NeGMA less responsive to modularity decreases (aligning with sGMA); (ii) Higher values of θ_Q enhance NeGMA sensitivity to modularity drops re-initialising more nodes (aligning with independent GMA); (iii) the memory term α has negligible or limited impact on the correctness and stability of αGMA.

5.4 Scalability Evaluation

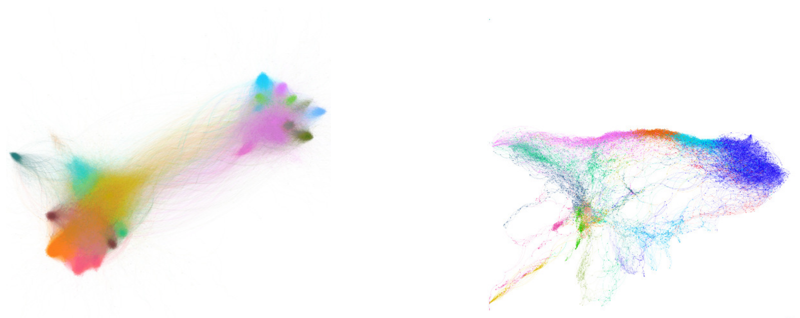
Finally, we evaluate the execution times of the algorithms as the graph evolves. αGMA exhibits convergence times $\approx 20\%$ slower than independent GMA. Preserving the past connections among nodes through α causes the algorithm to process a growing graph (more nodes and edges) at each snapshot. This impacts the overall scalability, making αGMA less suitable for constantly evolving dynamic graphs. Conversely, sGMA emerges as the most scalable algorithm compared to the baseline, with an execution time twice as fast. By retaining membership of the previously detected communities for existing nodes, it efficiently bootstraps the algorithm, yielding a sub-optimal solution as initialisation. Lastly, NeGMA achieves a convergence time comparable to the baseline.

Regarding the Leiden algorithm, it is noteworthy that only sLei demonstrates enhancement in convergence times, compared to independent Leiden, thereby conferring a significant advantage in scalability. However, for Leiden, the improvement in execution time is smaller, around 18%. In contrast, NeLei exhibits significantly slower convergence times than the baseline: unlike GMA, there is an average slowdown of 29%. Introducing a memory term in αLei consistently extended the convergence time, with an average delay of about 30%.

Takeaway: (i) αGMA is 20% slower than GMA due to added memory, making it less scalable for dynamic graphs. (ii) sGMA is twice as fast as the baseline, while NeGMA is similar to the baseline. (iii) sLei improves Leiden convergence by 18%, while NeLei slows down by 29%.

6 Evaluation in Real-World Graphs

In this section, we test the evolutionary extensions of the GMA algorithm to dynamic *real-world* Web-related scenarios. We focus on the GMA-based algorithms due to the method's performance trade-offs observed in the various synthetic scenarios analysed in Section 5, as well as due to the large use of GMA as a CD method applied to data gathered from the various layers of the Web (e.g., [5, 71, 74] *inter alia*).



(a) `VoterFraud` dataset. (b) `NetworkTelescope` dataset.

Fig. 9. Visualisation of a snapshot graph generated from the considered datasets with ForceAtlas2[36]. Different colours indicate communities detected by a single run of independent GMA.

We here rely on two well-known open datasets, namely `VoterFraud` and `NetworkTelescope`, which capture interactions among Twitter users and network scanning, respectively, using them to analyse the relative performance of the alternative algorithms in an unsupervised scenario. We selected these two real datasets because they are representative of two different communication layers of the Web, namely the lower network traffic layer and the application (social network) layer, which tend to exhibit very different evolutionary characteristics. On one hand, by construction, in the `NetwokTelescope` dataset, the embedding of the nodes gradually evolves toward a state of equilibrium, resulting in a sharp transformation of the graph over time. Thus, this dataset serves as a representative of cases where community structure shifts—characteristics that typically cannot be known beforehand and can only be hypothesized due to the unsupervised nature of the task. On the other hand, the `VoterFraud` dataset exemplifies cases of social media user analysis where activity is highly heterogeneous, producing a large amount of noise in the graph structure. Also, compared to the NetworkTelescope, we expect a much smoother community evolutionary process.

Notice that there is no ground truth in the datasets. Hence, unlike in the synthetic cases, here we cannot assess correctness or responsiveness. Thus, in addition to stability (which can be computed even in the absence of ground truth), we also consider modularity as a metric of the quality of the detected communities in each snapshot.

6.1 Datasets

6.1.1 `VoterFraud` *dataset.* This dataset [1, 2] encompasses a collection of Twitter posts containing keywords and hashtags related to the U.S. election fraud claims, spanning from 23-10-2020 to 16-12-2020. We build a weighted temporal retweet graph, where nodes are Twitter users and edges denote retweets between users. The edge weights indicate the number of retweets between two users. We aggregate data on a weekly basis obtaining 9 snapshots. At each snapshot, we filter out isolated nodes retaining only the giant weakly connected component (84% of the nodes). We show a summary of the resulting graph in Tab. 4.

Figure 9a provides a qualitative overview, showing a subgraph extracted from one snapshot, with colours representing communities detected in a single run of the independent GMA. Although two larger sides dominate, GMA identifies a more granular community structure.

This dataset exemplifies cases of social media user analysis where activity is highly heterogeneous producing a large amount of noise in the graph structure and the polarization of groups can vary over time, making this representative of a social network graph.

Table 4. Summary of graph metrics and averages per snapshot for real-dataset graphs: $|\mathcal{E}|$: overall number of edges; $|\mathcal{V}|$: overall number of nodes; N: number of snapshots; $\langle|\mathcal{E}^t|\rangle_t$: average number of edges over snapshots; $\langle|\mathcal{V}^t|\rangle_t$: average number of nodes over snapshots; $\langle \deg(v)\rangle_{v,t}$: average node degree across snapshots

	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{V}\rvert$	N	$\langle\lvert\mathcal{E}^t\rvert\rangle_t$	$\langle\lvert\mathcal{V}^t\rvert\rangle_t$	$\langle \deg(v)\rangle_{v,t}$
`VoterFraud`	~24M	~158k	9	~2.7M	~27k	437
`NetworkTelescope`	~900k	~1M	30	~35k	~13k	5

6.1.2 `NetworkTelescope` *dataset.* We here rely on a different aspect of the Web, i.e. raw packet traces collected by a network telescope. A network telescope consists of a set of passive IP addresses not hosting any services and receiving unsolicited traffic by definition [64]. This lack of production traffic makes such a network a privileged point of view for observing hosts running coordinated scans on the Internet (e.g. benign routine scans or distributed attacks performed by botnets [71]). This dataset [26, 27] encompasses one month of raw packet traces collected by a /24 network telescope hosted in our campus network during October 2022. We rely on host embeddings, i.e. numerical representations of hosts active in the considered network, to highlight similar scanning behaviours (we refer the reader to our previous works for methodological details [26, 27]). We build a weighted k-nearest-neighbours graph [24][8] on top of the embeddings and run CD algorithms to detect groups of coordinated hosts. We show a summary of the resulting dynamic graph in Table 4. By construction, the embeddings gradually evolve toward a state of equilibrium, resulting in a rapid transformation of the graph over time. Thus, this dataset is representative of the cases where community structure shifts — a characteristic that typically cannot be known beforehand and can only be hypothesized due to the unsupervised nature of the task.

In Figure 9b we provide a qualitative overview of one snapshot of the `NetworkTelescope` graph. In this case, distinct parts do not emerge as clearly as in the `VoterFraud` dataset and despite the comparable number of nodes per snapshot (i.e. tens of thousands), the lower number of edges (35k compared to 3M at each snapshot) results in a less dense graph and weaker community cohesion.

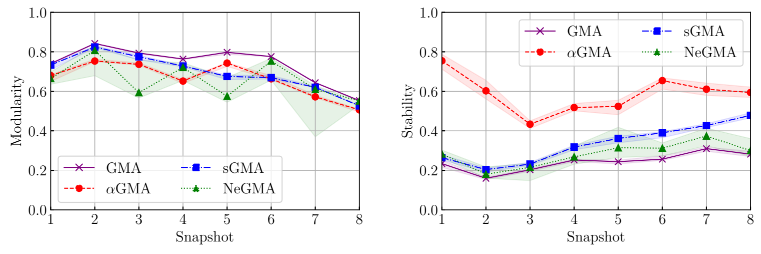
We emphasize that the two datasets offer diversity in our validation effort, capturing different aspects of the Web (i.e. interactions of users on a social network and traffic analysis at the packet level). The `NetworkTelescope` dataset exhibits a rapid and pronounced transformation of the graph over time, reflecting scenarios where the community structure evolves dynamically toward an equilibrium. In contrast, the `VoterFraud` dataset presents a smoother evolutionary process, typical of social media user activity characterized by high heterogeneity and substantial noise in the graph structure. This also impacts the meaning of the community: in the `VoterFraud` dataset, communities indicate political polarization (two larger communities), whereas in `NetworkTelescope` they highlight coordinated hosts scanning the same or complementary targets.

6.2 Experimental results

We now discuss modularity and stability results obtained running 50 instances of independent GMA and its three evolutionary extensions on the two real-world datasets. To run these experiments, we follow the results of the sensitivity analysis performed on the synthetic graphs discussed in Section 5.3 and set the modularity threshold for NeGMA as $\theta_Q = -0.01$ and the memory term of αGMA as $\alpha = 0.8$.

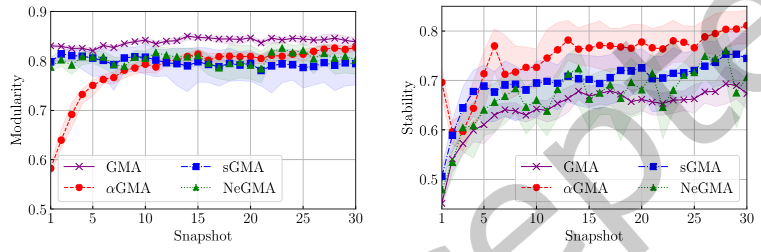
6.2.1 `VoterFraud` *dataset results.* We report modularity and stability in Figures 10a and 10b. Independent GMA, which is designed to yield independent solutions optimising modularity, achieves the highest-quality communities (≈ 0.8 of median modularity from snapshots 1 to 6). However, the three evolutionary algorithms demonstrate comparable modularity performance.

[8]In line with our previous works, we set k=3.



(a) Modularity. (b) Stability.

Fig. 10. Performance of the different algorithms with the VoterFraud dataset.



(a) Modularity. (b) Stability.

Fig. 11. Performance of the different algorithms with the NetworkTelescope dataset.

As expected, the good modularity performance achieved by GMA comes at the cost of temporal coherence. It exhibits notable instability (< 0.4 for all the snapshots, as seen in Figure 10b). sGMA marginally enhances stability compared to independent GMA, resulting in a maximum gain of $\approx 20\%$ in the last snapshot. At the same time, the parameter tuning of NeGMA on synthetic graphs allows it to find a trade-off between independent GMA and sGMA. Consistently, αGMA excels in stability, demonstrating per-snapshot improvements ranging from $\approx 50\%$ in snapshot 1 to $\approx 10\%$ in the last snapshot, outperforming other tested algorithms.

We notice that only 28% of nodes remain active in the network across multiple snapshots, which indicates sporadic user activity and may suggest that community dynamics are highly affected by intermittence-like transformations. Indeed, the good results obtained for αGMA in the VoterFraud dataset are consistent with those discussed in Section 5 and summarised in Table 2, which show that αGMA performs best with this transformation. Yet, as discussed in Section 5.4, the substantial improvement in the stability of αGMA greatly impacts the scalability in large graphs, with an average execution time of 892 seconds per snapshot compared to only 72 seconds taken by other algorithms.

We emphasise that CoDÆN enables faster and controlled comparisons of community detection algorithms. Avoiding the scalability challenges posed by large real-world graphs, it simplifies the selection of the best algorithm for specific requirements and helps in identifying optimal parameters for final applications.

6.2.2 `NetworkTelescope` *dataset results.* We report the modularity and stability at each snapshot in Figures 11a and 11b. We recall that, by construction, this is a much more dynamic graph, especially in the first snapshots ($t \in [1, 10]$), compared to both the VoterFraud and the synthetic graphs.

In light of this observation, while the memory term introduced of αGMA results in overall higher stability compared to the other algorithms (Figure 11b), it hurts the identification of changes in the graph, with a strong impact on modularity – i.e. it requires 10 snapshots to become comparable to sGMA and NeGMA (Figure 11a). On the other hand, independent GMA confirms the considerations of Section 5, i.e. it achieves poor stability, while leading to the highest modularity. sGMA, in turn, strikes a balance between independent GMA and αGMA both in terms of modularity (always > 0.75) and stability (≥ 0.70 for $t \geq 10$).

Interestingly, in contrast to the `VoterFraud` graph, where intermittence-like transformations seem to dominate, in `NetworkTelescope` the graph evolution is faster and much more complex, resulting in a highly dynamic node neighbourhood. This impacts NeGMA initialization, resulting in wide oscillations between the stability of independent GMA and sGMA.

All in all, the `NetworkTelescope` dataset offers a particularly challenging case study where the complexity of the graph dynamics has a strong impact on all methods, highlighting the limitation of introducing the memory term in the graph construction in terms of community quality (αGMA) and the struggles in evaluating a fast-changing neighbourhood in terms of stability (NeGMA).

Takeaway: (i) Evaluating real datasets without consistent ground truth is challenging, with no clear algorithmic winner for stability and community quality across scenarios; (ii) The results obtained on real data align with those from the synthetic graphs, i.e. αGMA is the most stable algorithm in cases when intermittent-like transformation dominates but suffers from low responsiveness in recognizing sudden changes. In turn, sGMA improves the stability of independent GMA, but detects suboptimal communities, while sGMA balances the two; (iii) CoDÆN addresses the scalability challenges of real-world dynamic graphs enabling faster comparison of alternative algorithms and guiding the parameter selection.

7 Conclusions

In this paper, we addressed the challenge of evaluating evolutionary CD algorithms in dynamic networks. We proposed a benchmarking framework (CoDÆN) relying on a set of graph transformations reflecting real-world scenarios. We supplied the lack of a comprehensive set of tools to assess the quality of detected communities proposing three new metrics (both supervised and unsupervised) complementary to the widely used modularity.

We adopted the proposed framework to extensively test and compare the performance of modularity-based evolutionary CD algorithms. We tested GMA, Leiden, and their evolutionary variants. We also proposed NeGMA (NeLei), a generalised modularity-based evolutionary CD approach relying on GMA (Leiden) and node neighbourhood. Additionally, we performed an investigation of the impacts of the parameters involved in the tested solutions. Finally, we tested the algorithms on real datasets related to the Web.

In a nutshell, our experimental findings reveal that (i) all the algorithms exhibit robustness against disruptive and noisy scenarios; (ii) while αGMA demonstrates both high stability and correctness in detecting intermittent transformations, it introduces high delays in detecting instantaneous transformations and strongly impacts convergence time; (iii) sGMA excels in scalability and maintains high stability over time (especially in gradual transformations affecting the community structure of the network), but exhibits low responsiveness and struggles in detecting emerging communities; (iv) NeGMA is the most balanced algorithm in terms of correctness and stability. Furthermore, it outperforms the responsiveness and detection rates of existing evolutionary solutions when abrupt changes occur. (v) performance on real datasets characterised by simple transformations reflects those on synthetic graphs, whereas (iv) more complex transformations highlight the limitations of the evolutionary CD algorithms.

Future developments of this work involve the extension of CoDÆN to more complex scenarios relying on combinations of individual transformations, along with the application of CoDÆN to other families of evolutionary CD algorithms not optimizing the modularity. Finally, we believe that sharing CoDÆN with the research community will promote its application on other datasets, possibly identifying strategies to choose the best approach given the characterised scenario.

Acknowledgments

This work has been funded by the project "National Center for HPC, Big Data and Quantum Computing", CN00000013 (Bando M42C – Investimento 1.4 – Avviso Centri Nazionali – D.D. n. 3138 of 16.12.2021, funded with MUR Decree n. 1031 of 17.06.2022), as well as by the CNPq (*Conselho Nacional de Desenvolvimento Científico e Tecnológico*) and FAPEMIG (*Fundação de Amparo à Pesquisa do Estado de Minas Gerais*) Brazilian funding agencies.

References

[1] Anton Abilov and Yiqing Hua. 2021. Voter Fraud 2020. (2021). doi:10.6084/m9.figshare.13571084.v2

[2] Anton Abilov, Yiqing Hua, Hana Matatov, Ofra Amir, and Mor Naaman. 2021. VoterFraud2020: a multi-modal dataset of election fraud claims on Twitter. *Proceedings of the International AAAI Conference on Web and Social Media* (2021). doi:10.1609/icwsm.v15i1.18113

[3] Huma Aftab, Junaid Shuja, Waleed Alasmary, and Eisa Alanazi. 2021. Hybrid DBSCAN based community detection for edge caching in social media applications. In *2021 International Wireless Communications and Mobile Computing (IWCMC)*.

[4] Mohsan Ali, Mehdi Hassan, Kashif Kifayat, Jin Young Kim, Saqib Hakak, and Muhammad Khurram Khan. 2023. Social media content classification and community detection using deep learning and graph analytics. *Technological Forecasting and Social Change* (2023).

[5] Giambattista Amati, Simone Angelini, Antonio Cruciani, Gianmarco Fusco, Giancarlo Gaudino, Daniele Pasquini, and Paola Vocca. 2021. Topic modeling by community detection algorithms. In *Proceedings of the 2021 Workshop on Open Challenges in Online Social Networks*.

[6] Abdelouahab Amira, Abdelouahid Derhab, Elmouatez Billah Karbab, and Omar Nouali. 2023. A survey of malware analysis using community detection algorithms. *Comput. Surveys* (2023).

[7] Thomas Aynaud and Jean-Loup Guillaume. 2010. Static community detection algorithms for evolving networks. *8th International Symposium on Modeling and Optimization in Mobile, ad hoc, and Wireless Networks.* https://ieeexplore.ieee.org/document/5520221/

[8] Albert-LaśzloB́arabaśi and Reḱa Albert. 1999. Emergence of Scaling in Random Networks. *Science* (1999). doi:10.1126/science.286.5439.509

[9] Alexis Baudin, Maximilien Danisch, Sergey Kirgizov, Clémence Magnien, and Marwan Ghanem. 2022. Clique percolation method: memory efficient almost exact communities. In *International Conference on Advanced Data Mining and Applications.*

[10] Alexis Baudin, Lionel Tabourier, and Clémence Magnien. 2023. LSCPM: Communities in Massive Real-World Link Streams by Clique Percolation Method. In *30th International Symposium on Temporal Representation and Reasoning, TIME 2023, September 25-26, 2023, NCSR Demokritos, Athens, Greece (LIPIcs, Vol. 278)*, Alexander Artikis, Florian Bruse, and Luke Hunsberger (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 3:1–3:18. doi:10.4230/LIPICS.TIME.2023.3

[11] Fabian Baumann, Philipp Lorenz-Spreen, Igor M Sokolov, and Michele Starnini. 2020. Modeling echo chambers and polarization dynamics in social networks. *Physical Review Letters* (2020). doi:10.1103/PhysRevLett.124.048301

[12] Punam Bedi and Chhavi Sharma. 2016. Community detection in social networks. *Wiley interdisciplinary reviews: Data mining and knowledge discovery* (2016).

[13] Ron Bekkerman, Shlomo Zilberstein, and James Allan. 2007. Web page clustering using heuristic search in the web graph. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence.*

[14] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment* (2008). doi:10.1088/1742-5468/2008/10/P10008

[15] Remy Cazabet, Souâad Boudebza, and Giulio Rossetti. 2020. Evaluating community detection algorithms for progressively evolving graphs. *Journal of Complex Networks* (2020). doi:10.1093/comnet/cnaa027

[16] Swarup Chattopadhyay, Tanmay Basu, Asit K Das, Kuntal Ghosh, and Late CA Murthy. 2021. Towards effective discovery of natural communities in complex networks and implications in e-commerce. *Electronic Commerce Research* (2021).

[17] Petr Chunaev. 2020. Community detection in node-attributed social networks: a survey. *Computer Science Review* (2020). doi:10.1016/j.cosrev.2020.100286

[18] Matteo Cinelli, Gianmarco De Francisci Morales, Alessandro Galeazzi, Walter Quattrociocchi, and Michele Starnini. 2021. The echo chamber effect on social media. *Proceedings of the National Academy of Sciences* 118, 9 (2021), e2023301118.

[19] Narimene Dakiche, Fatima Benbouzid-Si Tayeb, Yahya Slimani, and Karima Benatchba. 2019. Tracking community evolution in social networks: A survey. *Information Processing & Management* (2019). doi:10.1016/j.ipm.2018.03.005

[20] Lorenzo Dall'Amico, Romain Couillet, and Nicolas Tremblay. 2020. Optimal laplacian regularization for sparse spectral community detection. In *IEEE International Conference on Acoustics, Speech and Signal Processing.* doi:10.1109/ICASSP40776.2020.9053543

[21] Nur Nasuha Daud, Siti Hafizah Ab Hamid, Muntadher Saadoon, Firdaus Sahran, and Nor Badrul Anuar. 2020. Applications of link prediction in social networks: A review. *Journal of Network and Computer Applications* (2020). doi:10.1016/j.jnca.2020.102716

[22] Chi Thang Duong, Thanh Tam Nguyen, Trung-Dung Hoang, Hongzhi Yin, Matthias Weidlich, and Quoc Viet Hung Nguyen. 2023. Deep MinCut: Learning Node Embeddings by Detecting Communities. *Pattern Recognition* (2023).

[23] Heba Elgazzar, Kyle Spurlock, and Tanner Bogart. 2021. Evolutionary clustering and community detection algorithms for social media health surveillance. *Machine Learning with Applications* (2021). doi:10.1016/j.mlwa.2021.100084

[24] David Eppstein, Michael S Paterson, and F Frances Yao. 1997. On nearest-neighbor graphs. *Discrete & Computational Geometry* (1997).

[25] Sara E Garza and Satu Elisa Schaeffer. 2019. Community detection with the label propagation algorithm: a survey. *Physica A: Statistical Mechanics and its Applications* (2019).

[26] Luca Gioacchini, Luca Vassio, Marco Mellia, Idilio Drago, Zied Ben Houidi, and Dario Rossi. 2021. DarkVec: Automatic analysis of darknet traffic with word embeddings. In *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*.

[27] Luca Gioacchini, Luca Vassio, Marco Mellia, Idilio Drago, Zied Ben Houidi, and Dario Rossi. 2023. i-DarkVec: Incremental Embeddings for Darknet Traffic Analysis. *ACM Transactions on Internet Technology* (2023).

[28] Michelle Girvan and Mark EJ Newman. 2002. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences* (2002). doi:10.1073/pnas.122653799

[29] John W Goodell, Satish Kumar, Weng Marc Lim, and Debidutta Pattnaik. 2021. Artificial intelligence and machine learning in finance: Identifying foundations, themes, and research clusters from bibliometric analysis. *Journal of Behavioral and Experimental Finance* 32 (2021), 100577.

[30] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. 2018. Dyngem: Deep embedding method for dynamic graphs. *arXiv preprint arXiv:1805.11273* (2018).

[31] Chonghui Guo, Jiajia Wang, and Zhen Zhang. 2014. Evolutionary community structure discovery in dynamic weighted networks. *Physica A: Statistical Mechanics and its Applications* (2014).

[32] Khaled M Hammouda and Mohamed S Kamel. 2004. Efficient phrase-based document indexing for web document clustering. *IEEE Transactions on knowledge and data engineering* (2004).

[33] Yi Han, Shanika Karunasekera, and Christopher Leckie. 2020. Graph Neural Networks with Continual Learning for Fake News Detection from Social Media. arXiv:2007.03316

[34] Lukas Heumos, Anna C Schaar, Christopher Lance, Anastasia Litinetskaya, Felix Drost, Luke Zappia, Malte D Lücken, Daniel C Strobl, Juan Henao, Fabiola Curion, et al. 2023. Best practices for single-cell analysis across modalities. *Nature Reviews Genetics* 24, 8 (2023), 550–572.

[35] Sihan Huang, Haolei Weng, and Yang Feng. 2023. Spectral clustering via adaptive layer aggregation for multi-layer networks. *Journal of Computational and Graphical Statistics* (2023).

[36] Mathieu Jacomy, Tommaso Venturini, Sebastien Heymann, and Mathieu Bastian. 2014. ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software. *PLoS ONE* (2014). doi:10.1371/journal.pone.0098679

[37] Nikhil Jha, Martino Trevisan, Luca Vassio, and Marco Mellia. 2022. The Internet with Privacy Policies: Measuring The Web Upon Consent. *ACM Transactions on the Web* (2022). doi:10.1145/3555352

[38] Shengming Luo Jiashun Jin, Zheng Tracy Ke and Minzhe Wang. 2023. Optimal Estimation of the Number of Network Communities. *J. Amer. Statist. Assoc.* (2023). doi:10.1080/01621459.2022.2035736

[39] Di Jin, Zhizhi Yu, Pengfei Jiao, Shirui Pan, Dongxiao He, Jia Wu, S Yu Philip, and Weixiong Zhang. 2021. A survey of community detection approaches: From statistical modeling to deep learning. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[40] Michalis Kallitsis, Rupesh Prajapati, Vasant Honavar, Dinghao Wu, and John Yen. 2022. Detecting and Interpreting Changes in Scanning Behavior in Large Network Telescopes. *IEEE Transactions on Information Forensics and Security* (2022).

[41] ElMouatez Billah Karbab, Mourad Debbabi, Abdelouahid Derhab, and Djedjiga Mouheb. 2020. Android malware clustering using community detection on android packages similarity network. *arXiv preprint arXiv:2005.06075* (2020).

[42] ElMouatez Billah Karbab, Mourad Debbabi, Abdelouahid Derhab, and Djedjiga Mouheb. 2020. Scalable and robust unsupervised android malware fingerprinting using community-based network partitioning. *computers & security* (2020).

[43] Moaiad Ahmad Khder. 2021. Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application. *International Journal of Advances in Soft Computing & Its Applications* (2021). doi:10.15849/ijasca.211128.11

[44] Santosh Kumar and Ravi Kumar. 2021. A study on different aspects of web mining and research issues. *IOP conference series: Materials Science and Engineering*. doi:10.1088/1757-899X/1022/1/012018

[45] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. 2008. Benchmark graphs for testing community detection algorithms. *Physical Review E* (2008). doi:10.1103/PhysRevE.78.046110

[46] Chunying Li, Yong Tang, Zhikang Tang, Jinli Cao, and Yanchun Zhang. 2022. Motif-based embedding label propagation algorithm for community detection. *International Journal of Intelligent Systems* (2022).

[47] Zhao Li, Pengrui Hui, Peng Zhang, Jiaming Huang, Biao Wang, Ling Tian, Ji Zhang, Jianliang Gao, and Xing Tang. 2021. What happens behind the scene? Towards fraud community detection in e-commerce from online to offline. In *Companion Proceedings of the Web Conference 2021*.

[48] Yu-Ru Lin, Yun Chi, Shenghuo Zhu, Hari Sundaram, and Belle L Tseng. 2008. Facetnet: a framework for analyzing communities and their evolutions in dynamic networks. In *Proceedings of the 17th international conference on World Wide Web*. 685–694.
[49] Fuchen Liu, David Choi, Lu Xie, and Kathryn Roeder. 2018. Global spectral clustering in dynamic networks. *Proceedings of the National Academy of Sciences* 115, 5 (2018), 927–932.
[50] Fanzhen Liu, Zhao Li, Baokun Wang, Jia Wu, Jian Yang, Jiaming Huang, Yiqing Zhang, Weiqiang Wang, Shan Xue, Surya Nepal, et al. 2022. eRiskCom: an e-commerce risky community detection platform. *The VLDB Journal* (2022).
[51] Fanzhen Liu, Shan Xue, Jia Wu, Chuan Zhou, Wenbin Hu, Cecile Paris, Surya Nepal, Jian Yang, and Philip S Yu. 2021. Deep learning for community detection: progress, challenges and opportunities. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*. 4981–4987.
[52] Jingxin Liu, Chang Xu, Chang Yin, Weiqiang Wu, and You Song. 2020. K-core based temporal graph convolutional network for dynamic graphs. *IEEE Transactions on Knowledge and Data Engineering* 34, 8 (2020), 3841–3853.
[53] Yi-Ju Lu and Cheng-Te Li. 2020. GCAN: Graph-aware Co-Attention Networks for Explainable Fake News Detection on Social Media. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (Eds.). Association for Computational Linguistics, Online, 505–514. doi:10.18653/v1/2020.acl-main.48
[54] Mark EJ Newman. 2006. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences* (2006). doi:10.1073/pnas.0601602103
[55] Andrew Ng, Michael Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems* (2001). https://dl.acm.org/doi/10.5555/2980539.2980649
[56] Ahmed J Obaid, Tanusree Chatterjee, and Abhishek Bhattacharya. 2021. Semantic web and web page clustering algorithms: a landscape view. *EAI Endorsed Transactions on Energy Web* (2021).
[57] Gergely Palla, Albert-László Barabási, and Tamás Vicsek. 2007. Quantifying social group evolution. *Nature* 446, 7136 (2007), 664–667.
[58] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. 2005. Uncovering the overlapping community structure of complex networks in nature and society. *nature* (2005).
[59] Giordano Paoletti, Lorenzo Dall'Amico, Kyriaki Kalimeri, Jacopo Lenti, Yelena Mejova, Daniela Paolotti, Michele Starnini, and Michele Tizzani. 2024. Political context of the European vaccine debate on Twitter. *Scientific reports* 14, 1 (2024), 4397.
[60] Giordano Paoletti, Luca Gioacchini, Marco Mellia, Luca Vassio, and Jussara M. Almeida. 2023. Benchmarking Evolutionary Community Detection Algorithms in Dynamic Networks. https://doi.org/10.48550/arXiv.2312.13784 Presented at the *4th Workshop on Graphs and more Complex structures for Learning and Reasoning (GCLR) at AAAI 2024*.
[61] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. 2004. Defining and identifying communities in networks. *Proceedings of the national academy of sciences* 101, 9 (2004), 2658–2663.
[62] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. 2007. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E* (2007).
[63] Ebin Deni Raj, Gunasekaran Manogaran, Gautam Srivastava, and Yulei Wu. 2020. Information granulation-based community detection for social networks. *IEEE Transactions on Computational Social Systems* (2020). doi:10.1109/TCSS.2019.2963247
[64] Philipp Richter and Arthur Berger. 2019. Scanning the Scanners: Sensing the Internet from a Massively Distributed Network Telescope. In *Proceedings of the Internet Measurement Conference*. doi:10.1145/3355369.3355595
[65] Petar Ristoski and Heiko Paulheim. 2016. Semantic Web in data mining and knowledge discovery: A comprehensive survey. *Journal of Web Semantics* (2016). doi:10.1016/j.websem.2016.01.001
[66] Giulio Rossetti and Rémy Cazabet. 2018. Community Discovery in Dynamic Networks: A Survey. *Comput. Surveys* (2018). doi:10.1145/3172867
[67] Martin Rosvall and Carl T Bergstrom. 2008. Maps of random walks on complex networks reveal community structure. *Proceedings of the national academy of sciences* 105, 4 (2008), 1118–1123.
[68] Reza Saeedinia, S Omid Fatemi, Daniele Lorenzi, Farzad Tashtarian, and Christian Timmerer. 2023. Community-Based QoE Enhancement for User-Generated Content Live Streaming. In *2023 13th International Conference on Computer and Knowledge Engineering (ICCKE)*.
[69] Naw Safrin Sattar, Aydin Buluc, Khaled Z. Ibrahim, and Shaikh Arifuzzaman. 2023. Exploring temporal community evolution: algorithmic approaches and parallel optimization for dynamic community detection. *Applied Network Science* (2023). doi:10.1007/s41109-023-00592-1
[70] Adam Schenker, Mark Last, Horst Bunke, and Abraham Kandel. 2003. Clustering of web documents using a graph model. In *Web Document Analysis: Challenges and Opportunities*.
[71] Francesca Soro, Mauro Allegretta, Marco Mellia, Idilio Drago, and Leandro M Bertholdo. 2020. Sensing the noise: Uncovering communities in darknet traffic. In *2020 Mediterranean Communication and Computer Networking Conference (MedComNet)*.
[72] Myra Spiliopoulou, Eirini Ntoutsi, Yannis Theodoridis, and Rene Schult. 2013. MONIC and followups on modeling and monitoring cluster transitions. In *Machine Learning and Knowledge Discovery in Databases: European Conference*. doi:10.1007/978-3-642-40994-3_41
[73] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. 2019. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports* (2019). doi:10.1038/s41598-019-41695-z

[74] Hai Van Pham and Dong Nguyen Tien. 2021. Hybrid louvain-clustering model using knowledge graph for improvement of clustering user's behavior on social networks. In *Intelligent Systems and Networks: Selected Articles from ICISN.*

[75] Nguyen Xuan Vinh, Julien Epps, and James Bailey. 2009. Information theoretic measures for clusterings comparison: is a correction for chance necessary?. In *Proceedings of the 26th Annual International Conference on Machine Learning*. doi:10.1145/1553374.1553511

[76] Peizhuo Wang, Lin Gao, and Xiaoke Ma. 2017. Dynamic community detection based on network structural perturbation and topological similarity. *Journal of Statistical Mechanics: Theory and Experiment* 2017, 1 (2017), 013401.

[77] Yishu Wang, Ye Yuan, Yuliang Ma, and Guoren Wang. 2019. Time-dependent graphs: Definitions, applications, and algorithms. *Data Science and Engineering* (2019). doi:10.1007/s41019-019-00105-0

[78] Ling Wu, Qishan Zhang, Chi-Hua Chen, Kun Guo, and Deqin Wang. 2020. Deep learning techniques for community detection in social networks. *IEEE Access* (2020). doi:10.1109/ACCESS.2020.2996001

[79] Jaewon Yang and Jure Leskovec. 2012. Defining and evaluating network communities based on ground-truth. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*. 1–8.

[80] Ayijiang Yisimayi, Weiliang Song, Jing Wang, Fanchong Jian, Yuanling Yu, Xiaosu Chen, Yanli Xu, Sijie Yang, Xiao Niu, Tianhe Xiao, et al. 2024. Repeated Omicron exposures override ancestral SARS-CoV-2 immune imprinting. *Nature* 625, 7993 (2024), 148–156.

[81] Jiaxuan You, Tianyu Du, and Jure Leskovec. 2022. ROLAND: graph learning framework for dynamic graphs. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 2358–2366.

A Supplementary Material

Algorithm 1 Introducing memory to the graph (αGMA and αLei)

```
Require: Weighted dynamic graph $\{\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t)\}_{t=1}^N$, memory term $\alpha$
Ensure: Set of communities per snapshot $\{C^t\}_{t=1}^N$
1: for snapshot $t \leftarrow 1$ to $N$ do
2:     for edge $\epsilon = (u, v, \hat{w}^t) \in \mathcal{E}^t$ do
3:         if $w^{t-1} = 0$ then                    ▷ If the edge $\epsilon$ was not present in the previous snapshot
4:             $w^t \leftarrow \hat{w}^t$                          ▷ Preserve the edge weight
5:         else
6:             $w^t = (1 - \alpha) \cdot \hat{w}^t + \alpha \cdot w^{t-1}$          ▷ Embed the memory in the edge weight
7:         Update $\epsilon = (u, v, w^t) \in \mathcal{E}^t$          ▷ Replace the updated weight for the edge $\epsilon$
8:     $C^t \leftarrow$ IndependentAlg($\mathcal{V}^t, \mathcal{E}^t, \emptyset$)  ▷ Run CD algorithm on new graph with uninitialised communities
```

Algorithm 2 Introducing memory to the algorithm (sGMA and sLei).

```
Require: Weighted dynamic graph $\{\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t)\}_{t=1}^N$
Ensure: Set of communities per snapshot $\{C^t\}_{t=1}^N$
1: for snapshot $t \leftarrow 1$ to $N$ do
2:     if $t = 1$ then
3:         $\hat{C}^t \leftarrow \emptyset$                        ▷ Set the initialisation as an empty set
4:     else
5:         $\hat{C}^t \leftarrow C^{t-1}$      ▷ Set the GMA initialisation as the previously detected communities
6:     $C^t \leftarrow$ IndependentAlg($\mathcal{V}^t, \mathcal{E}^t, \hat{C}^t$)         ▷ Run CD algorithm with the custom initialisation
```

To facilitate comprehension and enhance the reproducibility of the research, this supplementary section provides pseudocode representations of each of the three evolutionary extensions investigated to integrate memory into the GMA and Leiden algorithms.

- Algorithm 1 - Memory in Edge Weights (αGMA and αLei): this algorithm integrates a memory parameter, α, to modulate edge weights contingent on their persistence across snapshots. In each snapshot t, the algorithm retains the present weight for edges that were absent in the previous step while assigning a weighted average of current and historical weights to existing edges. By inputting this adjusted graph into either GMA or Leiden, temporal consistency in community structures is preserved over time.
- Algorithm 2 - Memory in Initialization (sGMA and sLei): we modify the algorithm initialization by using the communities from the prior snapshot to initialize nodes, thus carrying forward community labels. Newly appearing nodes are assigned unique communities, while active nodes from the previous snapshot keep their community membership, allowing each algorithm to adjust as necessary for modularity optimization.
- Algorithm 3 - Neighborhood-based Initialization with Local Modularity Evaluation: The third methodology used herein involves the initialization of nodes depending on historical data and neighborhood context. Persistent nodes are assigned to their precedent communities, whereas new nodes adopt community memberships from neighbors. Subsequently, we conduct a modularity assessment to ascertain that these newly formed communities augment their local modularity. If the enhancement falls below the threshold

Algorithm 3 Overview of the proposed neighbourhood-based approach

```
1: function EvaluateModularity($\hat{C}^t, \hat{C}^{t-1}$)
2:     for community $c \in \hat{C}^{t-1}$ do                ▷ For each community in the provided initialisation
3:         $\Delta Q_c^t \leftarrow Q_c^t - Q_c^{t-1}$                    ▷ Get the local modularity gain
4:         if $\Delta Q_c^t < \theta_Q$ then                         ▷ If the local modularity degrades
5:             $\hat{C}^t \leftarrow \hat{C}^t - c$                        ▷ Unbind the community $c$
6:             $\hat{C}^t \leftarrow \hat{C}^t \bigcup_{v \in c}\{v\}$          ▷ Assign each node of $c$ to a distinct community
7:     return $\hat{C}^t$                                    ▷ Return the final initialisation

Require: Weighted dynamic graph $\{\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t)\}_{t=1}^N$, modularity threshold $\theta_Q$
Ensure: Set of communities per snapshot $\{C^t\}_{t=1}^N$
8: for snapshot $t \leftarrow 1$ to $N$ do
9:     $\hat{C}^t \leftarrow \emptyset$                               ▷ Set the initialisation as an empty set
10:    if $t > 1$ then
11:        for node $v \in \mathcal{V}^t$ do
12:            if $v \in \mathcal{V}^{t-1}$ then               ▷ If node $v$ was active is the previous snapshot
13:                $\hat{C}^t \leftarrow \hat{C}^t \bigcup c_v \in C^{t-1}$     ▷ Initialise $v$ with its previous community $c_v$
14:            else
15:                extract the active node neighbours $\mathcal{N}^{t(v)}$
16:                get the previous communities of the node neighbours $C^{t-1}_{\mathcal{N}^{t(v)}}$
17:                $\hat{C}^t \leftarrow \hat{C}^t \bigcup$ MajorityVoting($C^{t-1}_{\mathcal{N}}$)   ▷ Initialise $v$ with majority voting
18:        $\hat{C}^t \leftarrow$ EvaluateModularity($\hat{C}^t, \hat{C}^{t-1}$)            ▷ Evaluate local modularity
19:    $C^t \leftarrow$ IndependentAlg($\mathcal{V}^t, \mathcal{E}^t, \hat{C}^t$) ▷ Run independent GMA (or Leiden) with the custom initialisation
```

established in θ_Q, the nodes are reassigned to distinct communities prior to the final application of the community detection algorithm.

Received 29 January 2024; revised 26 November 2024; accepted 6 February 2025