POLITECNICO DI TORINO
Repository ISTITUZIONALE

Cultura del restauro in Piemonte tra Otto e Novecento e riflessi attuali. Dai Castelli di Rivara al Borgo Medievale, andata e ritorno / Restoration culture in Piedmont between the 19th

Original
Cultura del restauro in Piemonte tra Otto e Novecento e riflessi attuali. Dai Castelli di Rivara al Borgo Medievale, andata e ritorno / Restoration culture in Piedmont between the 19th and 20th centuries and current reflections. From the Castles of Rivara to the Borgo Medievale, round trip / Naretto, M.. - In: ATTI E RASSEGNA TECNICA. - ISSN 0004-7287. - STAMPA. - anno 157, LXXVIII:2-3(2024), pp. 20-33. [10.69100/A_RT.20240203.02]

Availability:
This version is available at: 11583/2995376 since: 2024-12-14T12:03:43Z

Publisher:
SIAT

Published
DOI:10.69100/A_RT.20240203.02

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

18 November 2026

Accepted at the 4th Workshop on Graphs and more Complex structures for Learning and Reasoning (GCLR) at AAAI 2024

arXiv:2312.13784v1 [cs.NE] 21 Dec 2023

Benchmarking Evolutionary Community Detection Algorithms in Dynamic Networks

Giordano Paoletti[1], Luca Gioacchini[1], Marco Mellia[1], Luca Vassio[1], Jussara M. Almeida[2]

[1]Politecnico di Torino, Turin, Italy
first.last@polito.it
[2]Universidade Federal de Minas Gerais, Belo Horizonte, Brazil
jussara@dcc.ufmg.br

Abstract

In dynamic complex networks, entities interact and form network communities that evolve over time. Among the many *static* Community Detection (CD) solutions, the modularity-based Louvain, or Greedy Modularity Algorithm (GMA), is widely employed in real-world applications due to its intuitiveness and scalability. Nevertheless, addressing CD in dynamic graphs remains an open problem, since the evolution of the network connections may poison the identification of communities, which may be evolving at a slower pace. Hence, naïvely applying GMA to successive network snapshots may lead to temporal inconsistencies in the communities. Two evolutionary adaptations of GMA, sGMA and αGMA, have been proposed to tackle this problem. Yet, evaluating the performance of these methods and understanding to which scenarios each one is better suited is challenging because of the lack of a comprehensive set of metrics and a consistent ground truth. To address these challenges, we propose (i) a benchmarking framework for evolutionary CD algorithms in dynamic networks and (ii) a generalised modularity-based approach (NeGMA). Our framework allows us to generate synthetic community-structured graphs and design evolving scenarios with nine basic graph transformations occurring at different rates. We evaluate performance through three metrics we define, i.e. Correctness, Delay, and Stability. Our findings reveal that αGMA is well-suited for detecting intermittent transformations, but struggles with abrupt changes; sGMA achieves superior stability, but fails to detect emerging communities; and NeGMA appears a well-balanced solution, excelling in responsiveness and instantaneous transformations detection.

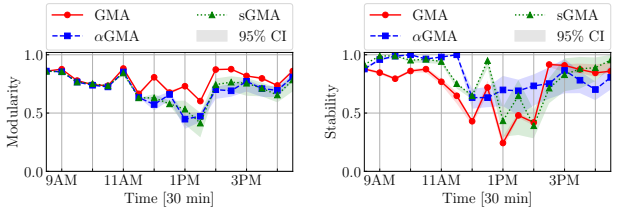
1 Introduction

Dynamic complex networks, or graphs, serve as a powerful tool to model computer, social or generic communications where entities (nodes) interact with each other establishing connections (edges) that evolve over time (Wang et al. 2019; Dakiche et al. 2019). Detecting communities, or groups of nodes that exhibit stronger internal connections compared to their connections with the broader network (Girvan and Newman 2002), and understanding their temporal dynamics is crucial for a deeper comprehension of the underlying structure and function of the systems they represent.

The scientific literature is rich in Community Detection (CD) approaches for static graphs, like the application of spectral clustering algorithms to the graph Laplacian (Ng, Jordan, and Weiss 2001; Dall'Amico, Couillet, and Tremblay 2020), hierarchical CD (Li et al. 2022) or statistical modelling (Peixoto 2019; Geng, Bhattacharya, and Pati 2019), just to cite few. Among them, the modularity-based Louvain, or Greedy Modularity Algorithm (GMA) (Blondel et al. 2008), is one of the most used methods for real-world applications. It is a fast and scalable solution to detect an unspecified number of communities in both small and large heterogeneous networks by optimising their *modularity* (Newman 2006), an intuitive and easily understandable measure of a graph tendency to form communities.

Nevertheless, due to the ever-changing nature of connections, network topologies are constantly changing. Such changes may or may not represent a natural evolution of the underlying communities. For example, short-term changes in node and edge sets might suggest substantial changes. However, beneath these rapid fluctuations, the underlying communities might remain consistent or evolve at a significantly slower pace than the observed changes suggest. Finding effective solutions to address CD problems in dynamic graphs is still an arduous challenge (Dakiche et al. 2019). A naïve approach involves independently running GMA on successive snapshots of the dynamic graph. Nevertheless, this necessitates an additional step to align communities across time, typically by comparing membership overlap (Spiliopoulou et al. 2013), which can result in a loss of temporal coherence. To our knowledge, the first attempt to adapt GMA for evolutionary scenarios is the Stabilised Louvain (sGMA) (Aynaud and Guillaume 2010). This method iteratively applies the standard Louvain over snapshots by initialising nodes as members of their previously assigned communities. Another approach is Louvain with memory (αGMA) (Elgazzar, Spurlock, and Bogart 2021). This approach incorporates a memory term into the weights of the graph edges prior to the application of standard GMA.

While these algorithms successfully maintain temporal consistency in community assignments, their relative performance in different scenarios of community evolution is still unclear. For illustration purposes, in Figure 1 we consider the detection of communities in the `SocioPatterns` dataset (Stehlé et al. 2011), which records the interactions between children and teachers in ten classes of a primary school during a day. To that end, we run 50 instances of independent GMA as well as the two aforementioned evolution-



(a) Modularity. (b) Algorithms stability.

Figure 1: Evaluation of static and evolutionary CD algorithms on the real-world `SocioPatterns` dataset.

ary extensions, comparing their performance with respect to graph modularity and stability – i.e. how similar the output is to the previous snapshot, measured every 30 minutes (see Section 2.3). The latter aims at capturing the consistency of communities over time, which is a desirable property in a realistic evolving scenario.

We observe two major effects: for stability, GMA tends to greatly change the communities across consecutive snapshots due to the natural randomness of contacts – i.e. in many snapshots, stability drops significantly, and both αGMA and sGMA improve it. Yet, GMA tends to offer better performance with respect to modularity, since it provides an independent and unconstrained solution for each snapshot. These results highlight the need for a trade-off between different community quality metrics in dynamic environments. Different CD approaches address such a trade-off in different ways. As such, it calls for a deeper study of the (relative) performance of evolutionary CD algorithms under different and controllable scenarios.

In this paper, we address these challenges by proposing a framework for benchmarking evolutionary modularity-based CD algorithms in dynamic networks. To evaluate alternative methods with respect to known ground truth, we generate community-structured graphs through the Lancichinetti-Fortunato-Radicchi (LFR) (Lancichinetti, Fortunato, and Radicchi 2008) benchmark and design 9 *in-vitro* graph transformations, reflecting evolutionary patterns that may arise in real-world scenarios and occurring at different rates. We evaluate the performance of evolutionary GMA solutions defining three metrics that capture complementary and yet desirable properties. Namely, *Correctness* and *Delay* to detect the transformations, which are supervised metrics leveraging the generated ground truth, and the aforementioned unsupervised *Stability* metric. To the best of our knowledge, we are the first proposing metrics specifically tailored for this goal.

Furthermore, we introduce a generalised evolutionary algorithm called Neighbourhood-based Louvain (NeGMA) and compare it against the existing αGMA and sGMA through the designed framework.

Our findings reveal that (i) αGMA achieves high stability in intermittent transformations, failing in detecting instantaneous transformations; (ii) sGMA outperforms all the tested solutions in terms of temporal coherence, but prevents the detection of new emerging communities; (iii) NeGMA is a well-balanced CD algorithm overcoming the limitation of existing solutions and proving high responsiveness and detection rates for instantaneous transformations.

2 Framework Overview

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[1]. 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[2], 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.

Given a set of non-overlapping communities C such that $\bigcup_{c \in C} c = \mathcal{V}$, we rely on the graph *modularity* (Newman 2006) to quantify 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 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 $Q = \sum_{c \in C} Q_c$.

GMA (Blondel et al. 2008) is designed to (greedily) maximise modularity.

2.1 Generating synthetic graphs

The evaluation of CD algorithms is often challenged by the lack of ground truth. It becomes even harder in dynamic scenarios when, as illustrated in Figure 1, metrics capturing different desired properties of a solution may produce quite opposite results. Thus, to conduct a sound and thorough evaluation of alternative evolutionary CD methods, we argue for the use of synthetic scenarios reflecting transformations in the graph topology that occur in real-world evolving networks and 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 (Lancichinetti, Fortunato, and Radicchi 2008), which is a widely recognised and versatile tool for creating complex networks exhibiting community structures and providing known ground-truth (i.e. the assigned communities). Any synthetic community-based graph generation algorithms can be used. LFR graphs are characterised by power-law degree distribution and heterogeneous community sizes. We randomly assign weights $\in (0, 1]$.

2.2 Simulated transformations and scenarios

Our framework creates 9 transformations, grouped into 3 scenarios – i.e. (i) Noise, (ii) Morphing and (iii) Disruptive.

[1] Possibly, each snapshot can represent a single edge or node change.

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

Table 1: Overview of scenarios and transformations.

Scenario	Transf.	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

These transformations can be applied at different rates. We provide a concise overview of the designed scenarios in Table 1.

Noise In this scenario we evaluate the CD algorithm's robustness and stability against perturbations over time. 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 while preserving node degree heterogeneity through preferential attachment (Barabaśi and Albert 1999). The new nodes persist throughout the simulation; (ii) *Intermittence*. At each snapshot t, we remove a fraction of nodes ϕ_{int} 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; and (iii) *Switch*. At each snapshot t, we remove a fraction of nodes ϕ_{swi}. Each node k is removed with probability p_k, as in the intermittence case. Inactive nodes reappear with probability $p_{swi} \propto \exp^{t_{off}-\gamma}$, where $\gamma \in \mathbb{N}$ is a user-defined parameter and t_{off} is the number of snapshots for which the node has been removed.

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

Morphing In this scenario we progressively change the community structure and the network topology over time. We start from the initial ground truth GT^0 for communities provided by the LFR and, after N snapshots, we end up with a different ground truth GT^N (hence different communities) which we define according to the considered transformation. We design 4 transformations: (i) *Split*. We select one or more communities and progressively split them into two sub-communities by loosening their edge weights; (ii) *Merge*. We select one or more community pairs and progressively merge them into a larger one by tightening the edge weights; (iii) *Death*. We select one (or more) communities and progressively dissolve them by loosening the intra-community edge weights and linking the nodes to other existing communities through preferential attachment; (iv) *Birth*. We progressively loosen the edge weights of some nodes per community and link them together forming 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, $w^t = \tau \pm w^{t-1}$. Notice that we clip $w^t \in (0,1]$.

These transformations reflect transitions in real-world networks, such as the rise of new communities in social networks centred on a specific topic or, conversely, the waning interest in certain subjects.

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 replacing the destination node of some edges with different existing nodes, preserving the weighted degree distribution; (ii) *Removal*. At each snapshot, we progressively remove a fraction of nodes ϕ_{rem} selected uniformly at random.

These transformations reflect real-world scenarios marked by a shift towards a disordered arrangement of nodes, like social mixing, where nodes from different communities randomly connect with each other.

2.3 Quality metrics

While modularity is the gold standard for evaluating community assignment, Figure 1 highlights that one single metric does not suffice for a complete evaluation of evolutionary cases. Thus, we define 3 metrics which capture complementary factors: *Stability* (unsupervised), *Correctness* and *Delay* (both use the information of the ground truth, hence supervised). These metrics rely on the Adjusted Mutual Information (AMI) (Vinh, Epps, and Bailey 2009), which measures the level of mutual information shared between two sets of communities accounting for unbalanced community sizes.

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

Correctness (K) Given a ground truth (GT), we evaluate the correctness of the considered 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 $\text{GT}=\text{GT}^0$. Additionally, we obtain the overall correctness K averaging K^t over the simulated snapshots.

Conversely, in the morphing scenario, the final ground truth is different from the initial one. Thus, we set $\text{GT}=\text{GT}^N$. Since we cannot define the final ground truth until the transformation has ended, we consider the overall correctness K as K^t at the final snapshot. Formally, $K = K^N$.

In disruptive transformations, the final ground truth GT^N is not available and we do not consider the correctness in this scenario.

Delay (D) It is the number of snapshots the algorithm takes to detect the change to a new community structure.

As such, 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, $\text{CP} = \min\{t : \text{AMI}(\text{GT}^N, C^t) > \text{AMI}(\text{GT}^0, C^t)\}$ Hence, we consider the delay D as the difference between the CP and the snapshot at which the transformation begins. If CP does not exist, we set the delay to the maximum possible value of t.

3 Modularity-based CD

The Louvain method, also known as the Greedy Modularity Algorithm (GMA) (Blondel et al. 2008), is a widely used and efficient approach for CD in static networks. It operates by iteratively greedily optimising the graph modularity. In a nutshell, at the first iteration, GMA assigns each node at random 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 invite the reader to refer to (Blondel et al. 2008) for details.

In the following, we consider evolutionary CD algorithms that use GMA as a starting point. A naïve evolutionary extension consists of applying GMA on each snapshot independently. We refer to this approach as *independent GMA*.

3.1 Louvain with memory (αGMA)

This approach (Elgazzar, Spurlock, and Bogart 2021) consists of prioritising the persistence of certain edges over time, favouring past partitions over new points to maintain community stability. In a nutshell, αGMA introduces a memory term α in the graph definition (notably in the edge weights) updating the network weights at snapshot t according to its history at snapshot $t-1$. Namely, an edge ϵ whose weight during snapshot t is $\hat{w}^t$ has its actual weight updated to $w^t = (1-\alpha)\cdot\hat{w}^t + \alpha\cdot w^{t-1}$ if $w^{t-1} \neq 0$. Otherwise, if $w^{t-1} = 0$, we set $w^t = \hat{w}^t$. The application of independent GMA on such a graph results in gradual modularity adjustments as new information becomes available.

3.2 Stabilised Louvain (sGMA)

This algorithm changes the initialisation process of GMA. Unlike the traditional method, which assigns each node to a unique distinct community, at snapshot t sGMA 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 GMA allowing nodes to adjust their community membership if the initial assignment is not suitable.

3.3 Novel Neighbourhood-based GMA (NeGMA)

Both αGMA and sGMA employ strategies to maintain stability in community assignment by keeping the new community structure as similar as possible to the previous snapshot. By doing so, both methods are expected to be better suited for detecting few or slow-rated modifications. Yet, they may struggle to properly track scenarios of greater or more abrupt changes. We here address this issue by (i) modifying the sGMA initialisation of new nodes which were inactive in the network relying on their neighbourhood in the current snapshot and (ii) evaluating the local community modularity Q_c after this novel initialisation. We call our approach NeGMA and provide an overview in Figure 2.

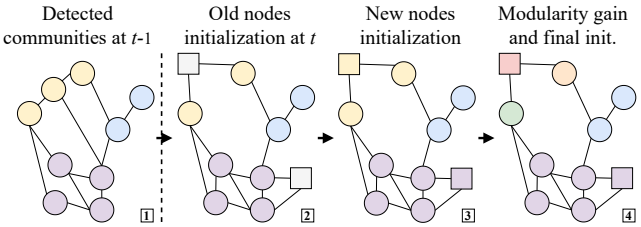


Figure 2: Proposed NeGMA initialisation of nodes at snapshot t starting from snapshot $t-1$. Different colours represent different communities; squares represent new nodes.

Namely, (i) we start from the communities detected at the previous snapshot $t-1$. At snapshot t the graph evolves, some new edges and nodes appear, and some disappear; (ii) for already existing nodes we assign them to the communities detected at $t-1$ (as in sGMA); (iii) for new nodes, we assign the community through majority voting on the communities of their *neighbourhood*; (iv) for each community, we compute the local modularity gain with respect to the ones of the previous snapshot: $\Delta Q_c^t = Q_c^t - Q_c^{t-1}$. If ΔQ_c^t is lower than a user-defined threshold θ_Q, we unbind the community assigning each of its nodes to a distinct community like traditional GMA (former yellow community in the upper-left part of the graph at step 4 in Figure 2).

4 Stability and Correctness Evaluation

We start our evaluation by assessing the stability and correctness of the CD algorithms under the various transformations presented in Table 1, deferring the discussion on delay to the next section. We run our experiments generating 100 independent graphs for each transformation. We set the LFR parameter $\mu = 0.2$ (Lancichinetti, Fortunato, and Radicchi 2008) and we generate $N = 150$ temporal snapshots. We trigger the beginning of the transformations at snapshot 25 and conclude it at snapshot 125. For each temporal graph, we perform 10 independent runs of each algorithm on each graph. To address the variability introduced by (i) the randomness of the algorithms and (ii) the graph generation, we initially average the results of the 10 runs on each graph. Consequently, we evaluate the median and bootstrapped 99% confidence interval for these medians across the 100 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 removal, we uniformly sample ϕ_{rem} between $[0.005, 0.02]$. When not explicitly stated, we set the transformation speed $\tau = 0.01$. As suggested by authors of αGMA (Elgazzar, Spurlock, and Bogart 2021), we set

Table 2: Overview of Correctness K and Stability S of the tested evolutionary algorithms compared to the median of the independent GMA (baseline). '=' indicates a performance variation $\leq 0.5\%$. A number n of '+' indicates an improvement in the $(\ 5(n-1)\%\ , 5n\%\ ]$ range. A number n of '–' indicates a decrease in the $[\ -5n\%\ , -5(n-1)\%\ )$ range.

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

† Recall that we compute differently the correctness for noise and morphing scenarios (see Section 2.3).

$\alpha = 0.8$, whereas for NeGMA we set $\theta_Q = 0$[3]. Finally, we run our experiments on a commodity server with 72 CPUs. We hope our framework and results can inspire other works towards the analysis of evolutionary CD. For that, we release our source code and generated graphs upon request.

In Table 2 we provide a bird's eye overview of the 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. For reference, we report also the values of the baseline.

Firstly, independent GMA achieves satisfactory performance in noisy scenarios (overall correctness $> 80\%$ and stability $\geq 90\%$). However, its stability is notably impacted by morphing transformations, dropping to 87% and 88% in birth and death transformations, respectively.

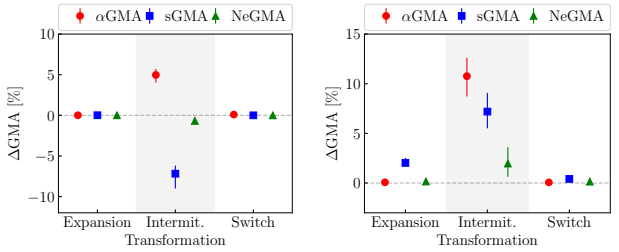
By preserving existing connections across successive snapshots, αGMA enhances correctness and, notably, stability under intermittence noise (both peaking at $\approx 100\%$), but delivers similar performance in all other transformations (with a small loss in stability for birth).

Both sGMA and NeGMA offer notable improvements in stability in scenarios when αGMA fails to do so – i.e. morphing transformations. Nevertheless, such improvements come at the expense of losses in correctness, especially for transformations leading to the emergence of new communities (i.e. split and birth). Yet, compared to sGMA, NeGMA offers a better trade-off between stability and correctness in such scenarios.

Finally, all the algorithms demonstrate remarkable stability (99%) performing on par with the baseline.

As argued, there is no single winner across all scenarios, and each algorithm offers a different trade-off between correctness and stability, depending on the transformation. Next, we delve deeper into these results and expand our discussion on how each algorithm behaves for these transformations.

[3]Notice that in case of modularity decrease for all the communities, NeGMA corresponds to independent GMA. We plan to evaluate the impact of θ_Q in future developments.



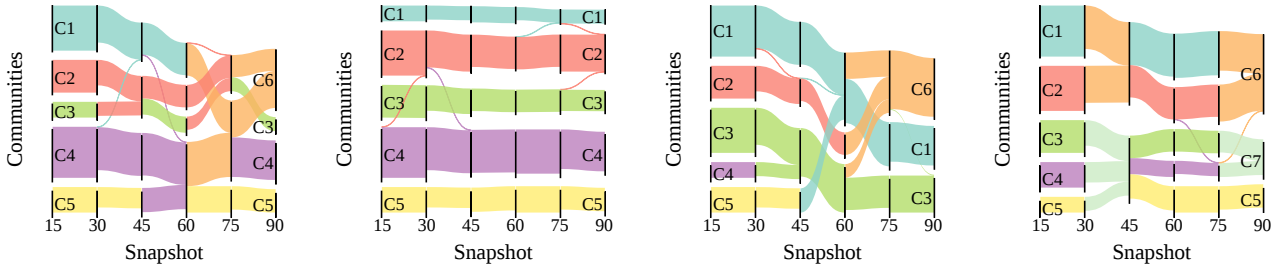
(a) Correctness (b) Stability

Figure 3: *Noise*. Median gain and bootstrapped 99% confidence interval in Correctness and Stability.

Noise transformations In Figure 3 we report *median* correctness and stability gain/loss compared to independent GMA in noise scenario, spanning from snapshots 25 to 125. All tested algorithms perform on par with GMA for the expansion and switch transformations, yielding a maximum gain of 2%.

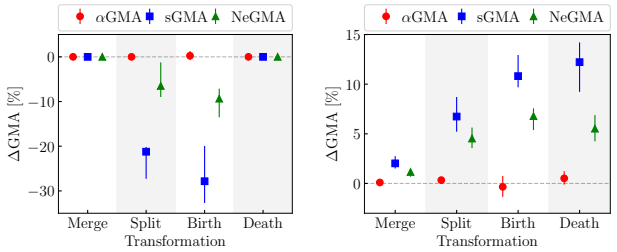
The memory term integrated into αGMA graph generation smooths the effects of the transformation, enhancing robustness against intermittent noise by boosting both correctness (+5%) and stability (+11%). This can be observed in the qualitative visualisation of the example in Figure 4b, where communities detected by αGMA at snapshot 15 remain largely consistent up to snapshot 90, despite minor fluctuations. In contrast, independent GMA suffers from constant changes in community structure and fails to return to the original configuration, as shown in Figure 4a.

sGMA initialisation improves the stability by 7% for intermittent transformation but leads to a 7% loss in correctness compared to independent GMA. As illustrated in the example of Figure 4c, the intermittent noise leads sGMA to merge communities at snapshot 60. Subsequent initialisation maintains these merged communities. Without a refinement, this causes a rearrangement of the existing communities preventing the detection of more communities in the next snapshot and ultimately the reconstruction of the original ones.



(a) GMA (b) αGMA (c) sGMA (d) NeGMA

Figure 4: 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.



(a) Correctness (b) Stability

Figure 5: *Morphing*. Median gain and bootstrapped 99% confidence interval in Correctness and Stability.

Conversely, NeGMA introduces a local modularity evaluation during initialisation. This overcomes the sGMA limitation and offers the best trade-off between correctness and stability. This results in a correctness decrease $< 1\%$ and a stability increase of $\approx 3\%$. Supporting this, Figure 4d illustrates partial reconstruction of original communities between snapshots 45 to 75.

Morphing transformations Figure 5 reports the *median* correctness and stability gain/loss resulting from morphing transformations. αGMA performs comparably to independent GMA across all designed transformations.

Notably, sGMA emerges as the top performer in terms of stability. It achieves a stability improvement ranging from +3% for merges, up to +13% for deaths. However, this improvement is countered by a significant loss of $> 20\%$ in correctness for split and birth transformations. This limitation is consistent with sGMA challenge in identifying emerging communities.

NeGMA once again proves to strike the best balance between correctness and stability. It achieves an overall maximum stability improvement of 7% while constraining the losses in correctness to no more than $\approx -10\%$ for both split and birth transformations.

Takeaway: (i) The memory term of αGMA does not bring substantial changes compared to independent GMA, a part from intermittence transformation; (ii) sGMA achieves high stability over time in all the scenarios, but fails in detecting emerging communities; (iii) NeGMA emerges as the most balanced algorithm in morphing transformations, achieving competitive performance in both correctness and stability.

Table 3: 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**

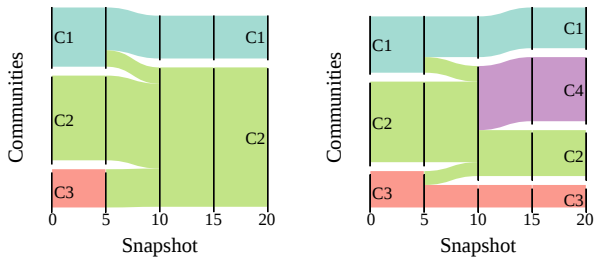
5 Responsiveness Evaluation

We assess the responsiveness of evolutionary GMA solutions in detecting changes through a new experiment involving 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 $t = 10$. We measure the delay D to detect this transformation. Given the setup, D falls in the $[0, 10]$ range and longer delays imply lower responsiveness.

Table 3 reports the median delay for each transformation and the fraction of detected transformations (CP is defined). 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).

While sGMA matches the delay of independent GMA for merges, births and deaths, it manages to reach only 4% of the CPs, failing to detect the vast majority (96%) of the splits (thus D is set to Max) due to the initialisation at each snapshot. Preserving the previous snapshot assignment, sGMA



(a) sGMA (b) NeGMA

Figure 6: Dynamic evolution of the graph with instantaneous birth transformation at time 10, showing the community assignment. Zoom in a sample of snapshots from 0 to 20.

fails to detect an increasing number of communities compared to the initialisation. It is worth noting that this limitation emerges also for birth transformations. Even though sGMA reaches 98% of CPs, it remains tied to the ones of GT^0 (as highlighted in Figure 6a) and it fails to detect emerging communities. This results in a correctness variation of -37.8% 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 (best performer for this transformation). Notably, even though NeGMA reaches 10% fewer CPs than sGMA, it successfully detects the transformations (unlike sGMA), as highlighted in Figure 6b, 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 insensitive to the abrupt formation of new communities; (iii) NeGMA exhibits high responsiveness and detection rates in most of the instantaneous morphing transformations.

6 Scalability Evaluation

Finally, we evaluate the algorithm execution times as the graph evolves gradually. 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 more edges) at each snapshot. This impacts the overall scalability making the 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. Exploring different modularity thresholds or employing a different community assignment of new nodes among their neighbourhood could further enhance NeGMA performance.

7 Conclusions

In this paper, we addressed the challenge of evaluating evolutionary CD algorithms in dynamic networks. We proposed a benchmarking framework relying on a set of graph transformations which reflect 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 also proposed NeGMA, a generalised modularity-based evolutionary CD approach relying on GMA and node neighbourhood. We adopted the proposed framework to extensively test and compare the performance of different evolutionary CD algorithms.

In a nutshell, experimental results 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 on 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.

Future developments include extending the designed transformations and testing different combinations of them including also real datasets. A promising extension of the proposed evaluation comprises other modularity-based CD approaches and their application to real-world problems. Additionally, a deeper investigation of the impacts of the parameters involved in the tested solutions can provide further insights into their performance, especially the modularity threshold θ_Q of NeGMA which could have a strong impact on some transformations, such as expansion.

References

Aynaud, T.; and Guillaume, J.-L. 2010. Static community detection algorithms for evolving networks. In *8th international symposium on modeling and optimization in mobile, ad hoc, and wireless networks*.

Barabaśi, A.-L.; and Albert, R. 1999. Emergence of Scaling in Random Networks. *Science*.

Blondel, V. D.; Guillaume, J.-L.; Lambiotte, R.; and Lefebvre, E. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*.

Dakiche, N.; Tayeb, F. B.-S.; Slimani, Y.; and Benatchba, K. 2019. Tracking community evolution in social networks: A survey. *Information Processing & Management*.

Dall'Amico, L.; Couillet, R.; and Tremblay, N. 2020. Optimal laplacian regularization for sparse spectral community detection. In *ICASSP 2020-2020 IEEE International*

Conference on Acoustics, Speech and Signal Processing (ICASSP), 3237–3241. IEEE.

Elgazzar, H.; Spurlock, K.; and Bogart, T. 2021. Evolutionary clustering and community detection algorithms for social media health surveillance. *Machine learning with applications*.

Geng, J.; Bhattacharya, A.; and Pati, D. 2019. Probabilistic community detection with unknown number of communities. *Journal of the American Statistical Association*, 114(526): 893–905.

Girvan, M.; and Newman, M. E. 2002. Community structure in social and biological networks. *Proceedings of the national academy of sciences*.

Lancichinetti, A.; Fortunato, S.; and Radicchi, F. 2008. Benchmark graphs for testing community detection algorithms. *Physical review E*.

Li, T.; Lei, L.; Bhattacharyya, S.; Van den Berge, K.; Sarkar, P.; Bickel, P. J.; and Levina, E. 2022. Hierarchical community detection by recursive partitioning. *Journal of the American Statistical Association*, 117(538): 951–968.

Newman, M. E. 2006. Modularity and community structure in networks. *Proceedings of the national academy of sciences*.

Ng, A.; Jordan, M.; and Weiss, Y. 2001. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14.

Peixoto, T. P. 2019. Network reconstruction and community detection from dynamics. *Physical review letters*, 123(12): 128301.

Spiliopoulou, M.; Ntoutsi, E.; Theodoridis, Y.; and Schult, R. 2013. MONIC and followups on modeling and monitoring cluster transitions. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13*.

Stehlé, J.; Voirin, N.; Barrat, A.; Cattuto, C.; Isella, L.; Pinton, J.-F.; Quaggiotto, M.; Van den Broeck, W.; Régis, C.; Lina, B.; et al. 2011. High-resolution measurements of face-to-face contact patterns in a primary school. *PloS one*.

Vinh, N. X.; Epps, J.; and Bailey, J. 2009. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th annual international conference on machine learning*.

Wang, Y.; Yuan, Y.; Ma, Y.; and Wang, G. 2019. Time-dependent graphs: Definitions, applications, and algorithms. *Data Science and Engineering*.