Benchmark model to assess community structure in evolving networks

Clara Granell,[1] Richard K. Darst,[2] Alex Arenas,[1,3] Santo Fortunato,[2] and Sergio Gómez[1]
[1]*Departament d'Enginyeria Informàtica i Matemàtiques, Universitat Rovira i Virgili, 43007 Tarragona, Spain*
[2]*Department of Computer Science, Aalto University School of Science, P.O. Box 12200, 00076 Aalto, Finland*
[3]*Institut Català de Paleoecologia Humana i Evolució Social, 43007 Tarragona, Spain*
(Received 23 January 2015; published 10 July 2015)

Detecting the time evolution of the community structure of networks is crucial to identify major changes in the internal organization of many complex systems, which may undergo important endogenous or exogenous events. This analysis can be done in two ways: considering each snapshot as an independent community detection problem or taking into account the whole evolution of the network. In the first case, one can apply static methods on the temporal snapshots, which correspond to configurations of the system in short time windows, and match afterward the communities across layers. Alternatively, one can develop dedicated dynamic procedures so that multiple snapshots are simultaneously taken into account while detecting communities, which allows us to keep memory of the flow. To check how well a method of any kind could capture the evolution of communities, suitable benchmarks are needed. Here we propose a model for generating simple dynamic benchmark graphs, based on stochastic block models. In them, the time evolution consists of a periodic oscillation of the system's structure between configurations with built-in community structure. We also propose the extension of quality comparison indices to the dynamic scenario.

DOI: 10.1103/PhysRevE.92.012805 PACS number(s): 89.75.Fb, 89.75.Hc

I. INTRODUCTION

The analysis and modeling of temporal networks has received a great deal of attention lately, mainly due to the increasing availability of time-stamped network data sets [1–5]. A relevant issue is whether and how the community structure of networks [6] changes in time. Communities reveal how networks are organized and function, hence major changes in their configuration might signal important turns in the evolution of the system as a whole, possibly anticipating dramatic developments such as rapid growth or disruption.

Indeed, there has been a great deal of activity around this topic in recent years [7–17]. However, most investigations lack strength on the validation part, which typically consists in checking whether the results of the algorithm make sense in one or more real networks whose community structure is usually unknown. Actually, it is not obvious what exactly it means to test an algorithm for detecting evolving communities. One idea could be that of correctly identifying the community structure of the system at each time stamp. However, during the evolution of the system several events that affect the network structure may occur, such as the creation or deletion of nodes or links or link rewiring, and it is not possible to detect these events by observing a single time-stamped network; they require taking into account the whole picture to be properly understood.

To explicitly keep track of the history of the system, an option is to consider multiple snapshots at once. For instance, in the evolutionary clustering approach [8] the goal is to find a partition that is descriptive of the structure of a given snapshot as well as correlated to the structure of the previous snapshots. Furthermore, the added value of any approach should be the ability to promptly detect changes in the community structure of the network. It would be possible to verify this if there were suitable benchmark graphs with evolving clusters, but those are still missing. This paper aims at filling this gap. We propose a model, derived from the classic stochastic block models [18–21], that generates three classes of dynamic benchmark graphs. The objective is to provide time-evolving networks such that at each snapshot the partition into communities is well defined according to the model. To keep things simple we consider a periodic evolution such that the same history repeats itself in cycles and is invariant under time reversal. The analysis of the community structure evolution for the designed benchmarks reveals that approaches exploiting the flow of system configurations might be more accurate in detecting the evolving community structure than methods that consider the snapshots independently. Note that in real data sets this evolution can be sharp and bursty, however in these cases the challenge of finding the community structure is not well defined because the range of time scales makes the mesoscopic structure clearly disconnected.

The paper is structured as follows. In Sec. II we describe the model to generate the benchmark networks. Section III introduces measures of comparison between dynamic clusterings. Section IV shows an example of the application of a dynamic multislice algorithm on the proposed benchmarks. Section V gives a summary and reports our conclusions.

II. MODEL DESCRIPTION

The model we propose for generating networks with evolving community structure is based on the classic stochastic block model (SBM) [18]. It works as follows. A network is divided into a number q of subgraphs and the nodes of the same subgraph are linked with a probability p_{in}, whereas nodes of different subgraphs are linked with a probability p_{out}. Such probabilities match the link densities within and between subgraphs. Supposing subgraphs of equal size, if $p_{\text{in}} > (q-1)p_{\text{out}}$ the resulting subgraphs are communities, as the (expected) link density within subgraphs exceeds their connectivity to the rest of the graph. The generation of samples from this model has a built-in efficiency: If there are $m_{\max}$ pairs of nodes, the actual number of edges is drawn from a binomial

distribution with parameters $m_{\max}$ and p. Then we simply place this number of edges randomly to generate a sample from our ensemble.

The model implements the two fundamental classes of dynamic processes: growing or shrinking and merging or splitting of communities. By combining these two reversible types of processes one can capture the most common behaviors of dynamic communities in real systems. We are then able to generate three standardized benchmarks. One consists in communities that grow and shrink in size (keeping fixed the total number of nodes of the network), while the second considers communities that merge and split. The third one is a mixed version of the previous two and consists of a combination of the last four operations.

A. Grow-shrink benchmark

This process models the movement of nodes from one community to another. At all times, two communities are kept in a SBM ensemble with intracommunity link density p_{in} and intercommunity link density p_{out}. However, the number of nodes in the two communities changes over time. In the basic process, we have a total of $2n$ nodes in two communities. In the balanced state, these are split into two equal communities of n nodes, which we call A and B. At the extremes, a fraction f of nodes in community A will switch to community B. If we take n_1 as the size of community A, then the number of nodes in the community B is $n_r = 2n - n_1$. Then, at time t the number of nodes in community A is

$$n_A = n - nf[2x(t + \tau/4) - 1], \tag{1}$$

with the $\tau/4$ phase factor specifying equal-sized communities at $t = 0$. The function $x(t)$ is the triangular waveform

$$x(t) = \begin{cases} 2t^*, & 0 \leqslant t^* < 1/2 \\ 2 - 2t^*, & 1/2 \leqslant t^* < 1 \end{cases} \tag{2}$$

[with $t^* \equiv (t/\tau + \phi) \bmod 1$], which controls the time periodicity. The constant ϕ is a phase factor with $\phi = 0$ for the $q = 2$ case and specified otherwise in the case of $q > 2$. With this formulation, we get communities of sizes (n,n), $(n - nf, n + nf)$, (n,n), and $(n + nf, n - nf)$ at $t/\tau \bmod 1 = 0, \frac{1}{4}, \frac{2}{4}$, and $\frac{3}{4}$, respectively. In practice, all $2n$ nodes are sorted in some arbitrary order and the first n_A nodes are put into community A and the others into community B. Say these nodes are $i = 0$ to $i = 2n - 1$.

After the community sizes are decided, the edges must be placed, taking into account that it is necessary that we keep the two communities in the proper SBM ensemble with equal and independent link probability p_{in} at all times. The independence of pairs provides a hint on how to do this. When a node j is moved from community A to B, all the existing edges of node j are removed. Then an edge is added between j and each node in the destination community B with equal and independent probability p_{in} and between j and each node in community A with equal and independent probability p_{out}, thus the ensemble is maintained. Conveniently, all edges can be precomputed and stored to allow a strictly repeating process, with the state at time t being identical to the state at time $t + \tau$, in analogy to the merging process.

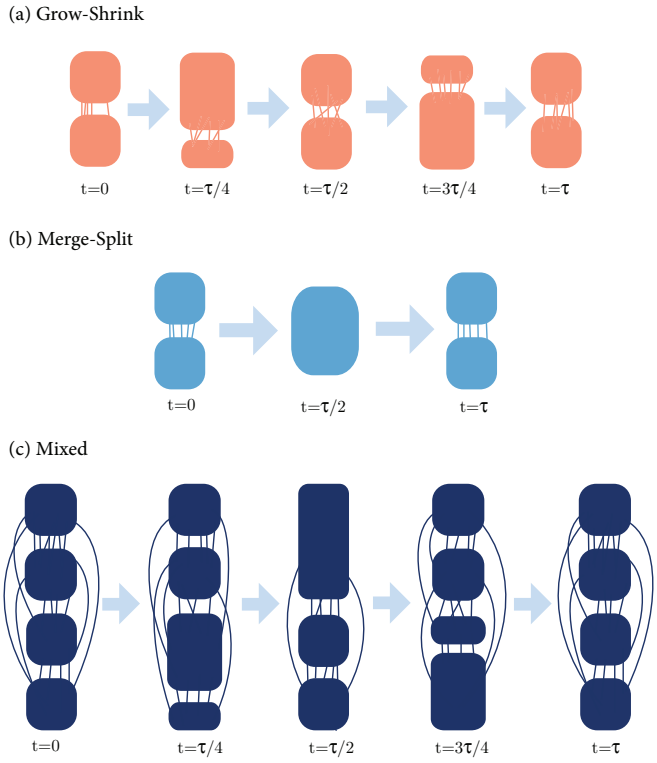

FIG. 1. (Color online) Schematic representation of the benchmarks. (a) Grow-shrink benchmark with $q = 2$. We begin with two equal-sized communities and over a period of τ nodes move from the bottom community to the top, then from the top to bottom, and then back to the symmetric state. (b) Merge-split benchmark with $q = 2$. We begin with two communities and over a period of τ we linearly add edges until there is one community with uniform link density and then reverse the process. (c) Mixed benchmark with $q = 4$, combining the merging and growing processes.

A special case that we need to cope with is the situation where f is very high and p_{in} is very low. When this happens, a community shrinks too much and it may become disconnected. In order to preserve the ensemble, we do not take actions to totally eliminate this possibility, but we ensure that $n(1 - f)p_{\text{in}} \gg 2$ to reduce the probability of disconnection. However, if a disconnection occurs, the process is aborted and rerun. Figure 1(a) is a sketch of the grow-shrink benchmark for the case $q = 2$.

B. Merge-split benchmark

This process models the merging of two communities. In this setup, we have a set of $2n$ nodes, divided into two communities of n nodes each. Each of the two initial communities has a link density of p_{in}, where those links are placed at initialization and kept unmodified over time. There are two extreme states: the unmerged and the merged state. In the unmerged state, all possible pairs of nodes between the two communities have an edge with probability p_{out}. This means that the network still has a connected component, but the nodes form two communities. In the merged state, all possible pairs of nodes between these two communities have an edge with probability p_{in}, which implies that all pairs of nodes in

the network have the same link density $p_{\rm in}$, the previous two communities are now indistinguishable, and thus we have one large community with $2n$ nodes.

The merge-split process is a periodic interpolation of the merged and unmerged states. The numbers of intercommunity edges in the unmerged state $m_{\rm um}$ and in the merged state $m_{\rm m}$ are first picked from a binomial distribution consistent with the binomial distribution parameters n^2 and $p_{\rm out}$ or $p_{\rm in}$. All possible intercommunity edges are placed in some arbitrary but random order and the first

$$m^*(t) = [1 - x(t)]m_{\rm um} + x(t)m_{\rm m} \tag{3}$$

edges are selected to be active at time t. The effective intracommunity link density is $p^*_{\rm inter}(t) = m^*(t)/n^2$. The parameter $x(t)$ is the triangular waveform from Eq. (2). In practice, this means that at time $t/\tau \mod 1 = 0$ the communities are unmerged and at $t/\tau \mod 1 = 1/2$ the communities are merged, with linear interpolation (of the number of edges) between these points. Since the possible edges are ordered only at initialization, the process is strictly periodic, that is, the edges present at time t are identical to those present at time $t + \tau$.

One may think that the communities are fully merged at the extreme of this process, where the intercommunity link density is $p^*_{\rm inter} = p_{\rm in}$ (at $t = \tau/2$). However, due to the *detectability limit* of communities in stochastic block models, this is not the case [22]. Even when $p_{\rm out} < p_{\rm in}$, it can be that the configuration is indistinguishable from one large community. Following [22], at the point

$$p_{\rm in} - p^*_{\rm inter} = \sqrt{\frac{1}{n}(p_{\rm in} + p^*_{\rm inter})} \tag{4}$$

we consider the communities to be merged into one for all practical purposes. While this limit is strictly speaking only accurate in the sparse and infinite-size limit, it is an adequate approximation. A schematic representation of the merge-split benchmark for $q = 2$ is shown in Fig. 1(b).

C. Mixed benchmark

This process is a combination of the merging and growing processes. In this process, there is a total of $4n$ nodes with two merging-splitting communities ($2n$ nodes) and two growing-shrinking communities ($2n$ nodes). The intracommunity links are managed with the same processes as above with phase factors of $\phi = 0$ for both. If there are $q = 4a > 4$ total communities, then the pairs of communities involved in merging and growing process have phase factors $\phi = 0, \frac{1}{a}, \frac{2}{a}, \ldots, \frac{a-1}{a}$. Between the pairs of nodes that belong to different processes, an edge exists with a probability of $p_{\rm out}$. Figure 1(c) exemplifies the mixed benchmark when $q = 4$.

III. TIME-DEPENDENT COMPARISON MEASURES

The assessment of the performance of any clustering algorithm requires the use of measures to define the distance or similarity between any pair of partitions. The list of available measures is long, including, e.g., the Jaccard index [23], the Rand index [24], the adjusted Rand index [25], the normalized mutual information [26], the van Dongen metric [27], and the normalized variation of information metric [28]. All of them have in common the possibility of being expressed in terms of the elements of the so-called confusion matrix or contingency table, thus we focus first on its calculation. Let $\mathcal{C} = \{C_\alpha | \alpha = 1, \ldots, r\}$ and $\mathcal{C}' = \{C'_{\alpha'} | \alpha' = 1, \ldots, r'\}$ be two partitions of the data in r and r' disjoint clusters. The $\alpha\alpha'$th component of the contingency table M accounts for the number of elements in the intersection of clusters C_α and $C'_{\alpha'}$,

$$m_{\alpha\alpha'} = |C_\alpha \cap C'_{\alpha'}|. \tag{5}$$

The sizes of the clusters simply read $n_\alpha = |C_\alpha| = \sum_{\alpha'} m_{\alpha\alpha'}$ and $n'_{\alpha'} = |C'_{\alpha'}| = \sum_\alpha m_{\alpha\alpha'}$ and the total number of elements is $N = \sum_\alpha n_\alpha = \sum_{\alpha'} n'_{\alpha'} = \sum_\alpha \sum_{\alpha'} m_{\alpha\alpha'}$. With these definitions at hand, one can calculate the Jaccard index

$$J = \frac{\sum_\alpha \sum_{\alpha'} \binom{m_{\alpha\alpha'}}{2}}{\sum_\alpha \binom{n_\alpha}{2} + \sum_{\alpha'} \binom{n'_{\alpha'}}{2} - \sum_\alpha \sum_{\alpha'} \binom{m_{\alpha\alpha'}}{2}}, \tag{6}$$

the normalized mutual information (NMI) index

$$\mathrm{NMI} = \frac{-2 \sum_\alpha \sum_{\alpha'} m_{\alpha\alpha'} \log\left(N m_{\alpha\alpha'}/n_\alpha n'_{\alpha'}\right)}{\sum_\alpha n_\alpha \log\left(n_\alpha/N\right) + \sum_{\alpha'} n'_{\alpha'} \log\left(n'_{\alpha'}/N\right)}, \tag{7}$$

and the normalized variation of information (NVI) metric

$$\mathrm{NVI} = \frac{-1}{\log N} \sum_\alpha \sum_{\alpha'} \frac{m_{\alpha\alpha'}}{N} \log \frac{(m_{\alpha\alpha'})^2}{n_\alpha n'_{\alpha'}}, \tag{8}$$

where, by convention, $0 \log 0 = 0$.

In the case of evolving networks we have to compare two sequences of partitions $\{\mathcal{C}(t) | t = 1, \ldots, T\}$ and $\{\mathcal{C}'(t) | t = 1, \ldots, T\}$, a task that can be performed in different ways. The simplest solution is the independent comparison of partitions at each time step by measuring the similarity or distance between $\mathcal{C}(t)$ and $\mathcal{C}'(t)$ for each value of t, thus obtaining, e.g., a Jaccard index $J(t)$ for each snapshot [see Fig. 2(a)]. However, this procedure discards the evolutionary nature of the communities: We would like to quantify not only the static resemblance of the communities but also if they evolve in a similar way.

Our proposal consists in the definition of windowed forms of the different indices and metrics, obtained by considering sequences of consecutive partitions, i.e., time windows of a predefined duration σ. In Fig. 2(b) we show the comparison between individual snapshots and sequences of length 2. For example, let us consider the time window formed by time steps from t to $t + \sigma$. Every node belongs to a different cluster at each snapshot and this evolution can be identified as one of the items in $\mathcal{D}(t;\sigma) = \mathcal{C}(t) \times \mathcal{C}(t+1) \times \cdots \times \mathcal{C}(t+\sigma)$ for the first sequence of partitions and $\mathcal{D}'(t;\sigma) = \mathcal{C}'(t) \times \cdots \times \mathcal{C}'(t+\sigma)$ for the second one, where the multiplication sign denotes the Cartesian product of sets. Since the number of nodes is N, there are at most N different nonvoid sets $D_\alpha(t;\sigma) \in \mathcal{D}(t;\sigma)$ and the same for $D'_{\alpha'}(t;\sigma) \in \mathcal{D}'(t;\sigma)$. For example, in Fig. 2(b), the combinations of partitions (excluding empty sets) are $\mathcal{D}(t = 1; \sigma = 1) = \{\mathrm{AA}, \mathrm{AB}, \mathrm{BB}, \mathrm{CC}\}$ and $\mathcal{D}'(t = 1; \sigma = 1) = \{\mathrm{AA}, \mathrm{BB}, \mathrm{CC}\}$. Next, we may define the elements of the contingency table for this time window as

$$m_{\alpha\alpha'}(t;\sigma) = |D_\alpha(t;\sigma) \cap D'_{\alpha'}(t;\sigma)|, \tag{9}$$

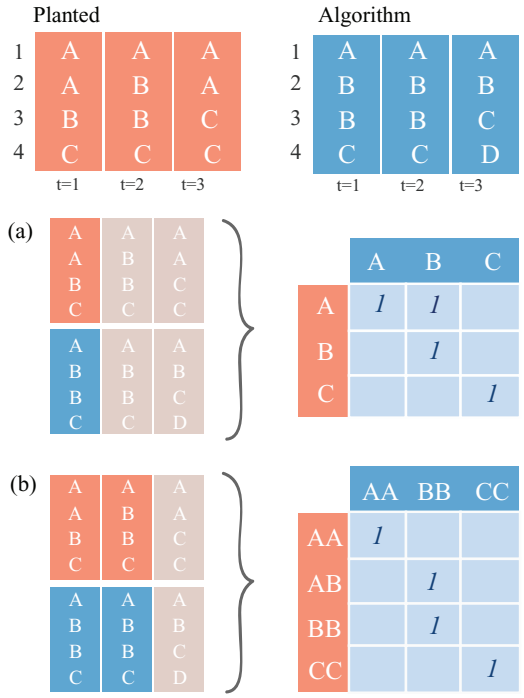

FIG. 2. (Color online) Construction of the contingency tables $m_{\alpha\alpha'}$. On top we represent three steps (columns) of the time evolution of a network of four nodes (rows) and the partitions in communities we want to compare, e.g., the planted partitions from the benchmark and those obtained by a certain algorithm. To compare these two partitionings, we can do it as it is depicted in (a), which takes only one snapshot at a time ($\sigma = 0$), or as in (b), building a contingency table where the entries consider two snapshots at the same time ($\sigma = 1$). Afterward, the measures (NVI, NMI, or Jaccard index) are calculated from these tables.

which accounts for the number of nodes following the same cluster evolutions $D_\alpha(t;\sigma)$ and $D'_{\alpha'}(t;\sigma)$. Likewise, we have

$$n_\alpha(t;\sigma) = |D_\alpha(t;\sigma)| = \sum_{\alpha'} m_{\alpha\alpha'}(t;\sigma), \tag{10}$$

$$n'_{\alpha'}(t;\sigma) = |D'_{\alpha'}(t;\sigma)| = \sum_{\alpha} m_{\alpha\alpha'}(t;\sigma), \tag{11}$$

and

$$N = \sum_\alpha n_\alpha(t;\sigma) = \sum_{\alpha'} n'_{\alpha'}(t;\sigma) = \sum_\alpha \sum_{\alpha'} m_{\alpha\alpha'}(t;\sigma). \tag{12}$$

Finally, we may use Eqs. (6)–(8) to calculate the corresponding windowed Jaccard index $J(t;\sigma)$, windowed normalized mutual information index $\mathrm{NMI}(t;\sigma)$, and windowed normalized variation of information metric $\mathrm{NVI}(t;\sigma)$, respectively. Of course, the windowed measures reduce to the standard static ones when $\sigma = 0$ and are able to capture differences in the evolution of communities that cannot be distinguished using their classical versions (see the Appendix).

We will see in the next section how the plots of $\mathrm{NVI}(t;\sigma)$ are valuable to compare different algorithms and to detect in which moments of the time evolution they differ. Nevertheless, it is also convenient to have a single number to quantify the overall deviation. A simple solution is the use of the average-squared errors, which is expressed as follows:

$$E_{\mathrm{J}}(\sigma) = \frac{1}{T}\sum_{t=1}^{T}[J(t;\sigma) - 1]^2, \tag{13}$$

$$E_{\mathrm{NMI}}(\sigma) = \frac{1}{T}\sum_{t=1}^{T}[\mathrm{NMI}(t;\sigma) - 1]^2, \tag{14}$$

$$E_{\mathrm{NVI}}(\sigma) = \frac{1}{T}\sum_{t=1}^{T}\mathrm{NVI}(t;\sigma)^2. \tag{15}$$

For simplicity and for its superior mathematical properties (see [28]) we have chosen to use only the NVI metrics in the rest of this article. See Ref. [29] for the results using the normalized mutual information and the Jaccard index.

IV. RESULTS

Here we show an example of the application of a community detection algorithm, designed to take into account the evolution of complex networks, to reveal the community structure in our benchmarks. The chosen method is the multislice algorithm in [12], which extends the definition of modularity to multilayer networks. In their representation, each layer (slice) consists of a single network at a particular time. The slices are connected between them by joining each node with its counterpart in the next and previous layer and this link has a specified weight ω, equal for all links of this kind, which acts as a tuning parameter. For $\omega = 0$, no connection between slices is considered and the algorithm is performed statically. As this value increases, more consideration is given to the communities across layers. The formulation includes an additional parameter γ, which accounts for the tuning of the resolution at which communities are found, in the manner of [30]. In this work, we have used the code available in [31], setting the resolution parameter γ to 1 and varying the interslice coupling ω.

The benchmarks used to put to test this algorithm are generated using the model proposed in this paper. For the sake of simplicity, we generate three simple standard benchmarks, one for each basic procedure: grow-shrink, merge-split, and mixed. The grow-shrink benchmark consists in a network with $q = 2$ communities, where each community has initially $n = 32$ nodes (therefore the total size of the network is $N = 64$), with $p_{\mathrm{in}} = 0.5$, $p_{\mathrm{out}} = 0.05$, $f = 0.5$, and $\tau = 100$ time steps. The merge-split test has a variable number of communities; in this paper we use the parameters $q = 2$ communities of size $n = 32$ each, with $p_{\mathrm{in}} = 0.5$, $p_{\mathrm{out}} = 0.05$, and $\tau = 100$. The mixed benchmark, a combination of the previous two, has $q = 4$ communities of $n = 32$ nodes each and the other parameters are set as in the previous cases.

Figure 3 shows the planted partitions for the three benchmarks and the results from the multislice algorithm at three different interslice couplings: In the extreme case $\omega = 0$ slices are considered independently, $\omega = 0.5$ is an intermediate value that provides good results, and $\omega = 2$ provides an example of the partitioning obtained when using strong coupling between layers. It can be seen that for $\omega = 0$ we obtain a different partition for each time step and the results are mostly correct, except for those configurations of the sizes of the

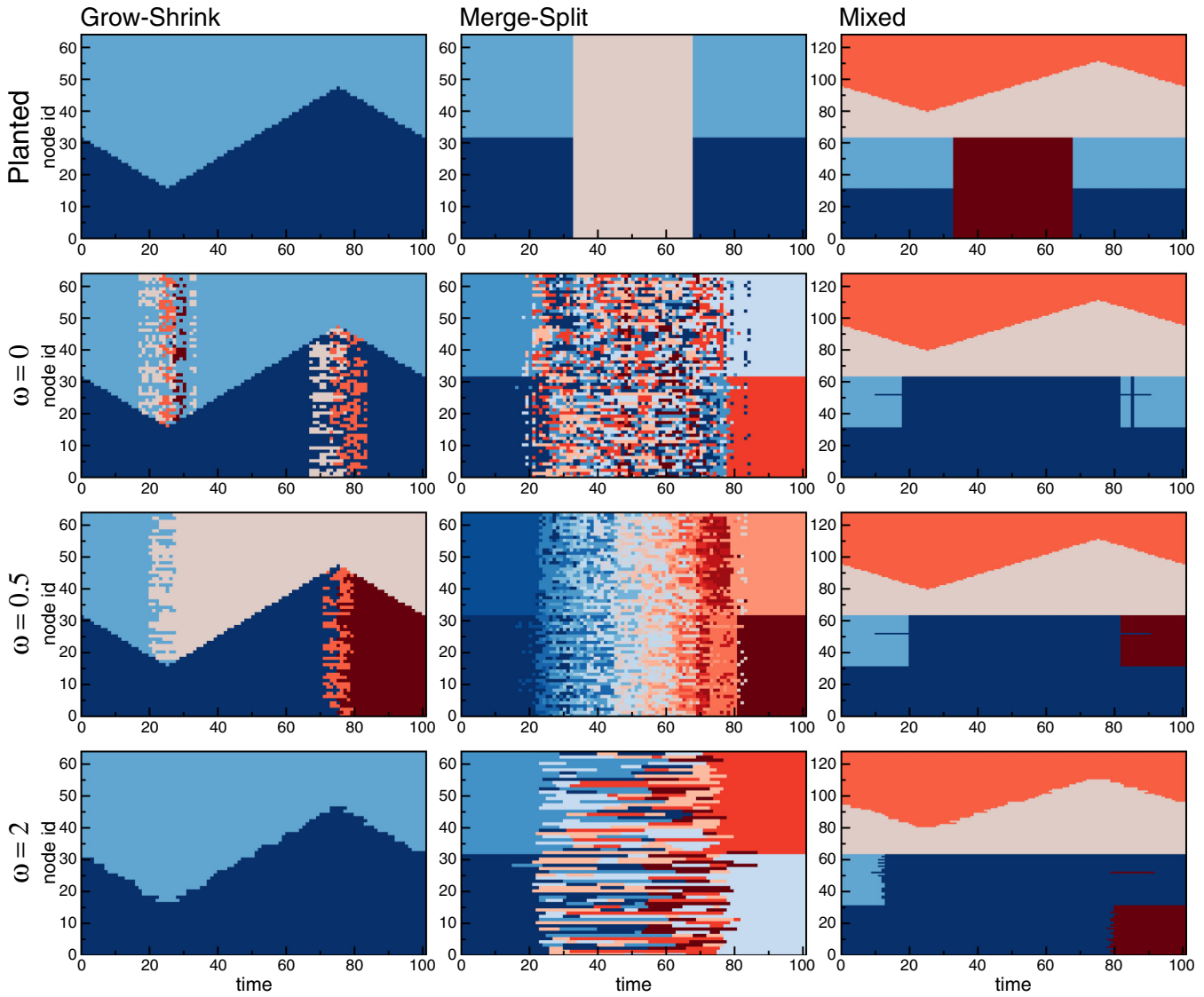

FIG. 3. (Color online) Results of the application of the multislice community detection method to the three benchmarks proposed (in columns). The first row corresponds to the planted partition of each benchmark, while the three remaining rows are the partitions obtained by the multislice algorithm for different values of the interslice parameter ω, which is the weight of the coupling between different instances of the same nodes across layers. When $\omega = 0$ the slices are disconnected and then the community detection analysis is done for each slice separately. As this value increases, more importance is given to the evolving nature of the problem and communities across slices are found. In each plot, the vertical axis corresponds to the index of nodes in the network, while the horizontal axis represents the time. The color of each pair {node, time} is the label of the community at which the node is assigned at that specific time.

communities where the preference of modularity for equal-sized communities hampers the process (see the first column of Fig. 3). Higher values of ω request higher consistency through time, which implies that the number of misclassified individual snapshots is reduced. We have also compared the multislice method with a temporal stability approach [32] and the results obtained are very similar to the results of the multislice algorithm obtained at $\omega = 0.5$.

To quantitatively evaluate the results, we use the windowed measures introduced in the previous section. We calculate the measures between the partitions obtained by the algorithm and the planted ones, for three values of the time window. When the time window is of size 1 ($\sigma = 0$), each snapshot is considered independently, that is, we have computed the measure between the planted partition at t and the algorithm's result at t, repeating this process until $t = \tau$. Instead, with the time window of size 2 ($\sigma = 1$), we evaluate the evolution of the partitions during two consecutive time steps, following the same process but comparing the planted partitions at $[t,t+1]$ with the algorithm's results at $[t,t+1]$. This formulation is more restrictive, as we impose, in addition to the condition that the nodes must belong to the same community, that their evolution during two consecutive time steps is also the same. Similarly, we have also analyzed time windows of size 5 ($\sigma = 4$) to check the quality of the detected community evolutions at longer ranges.

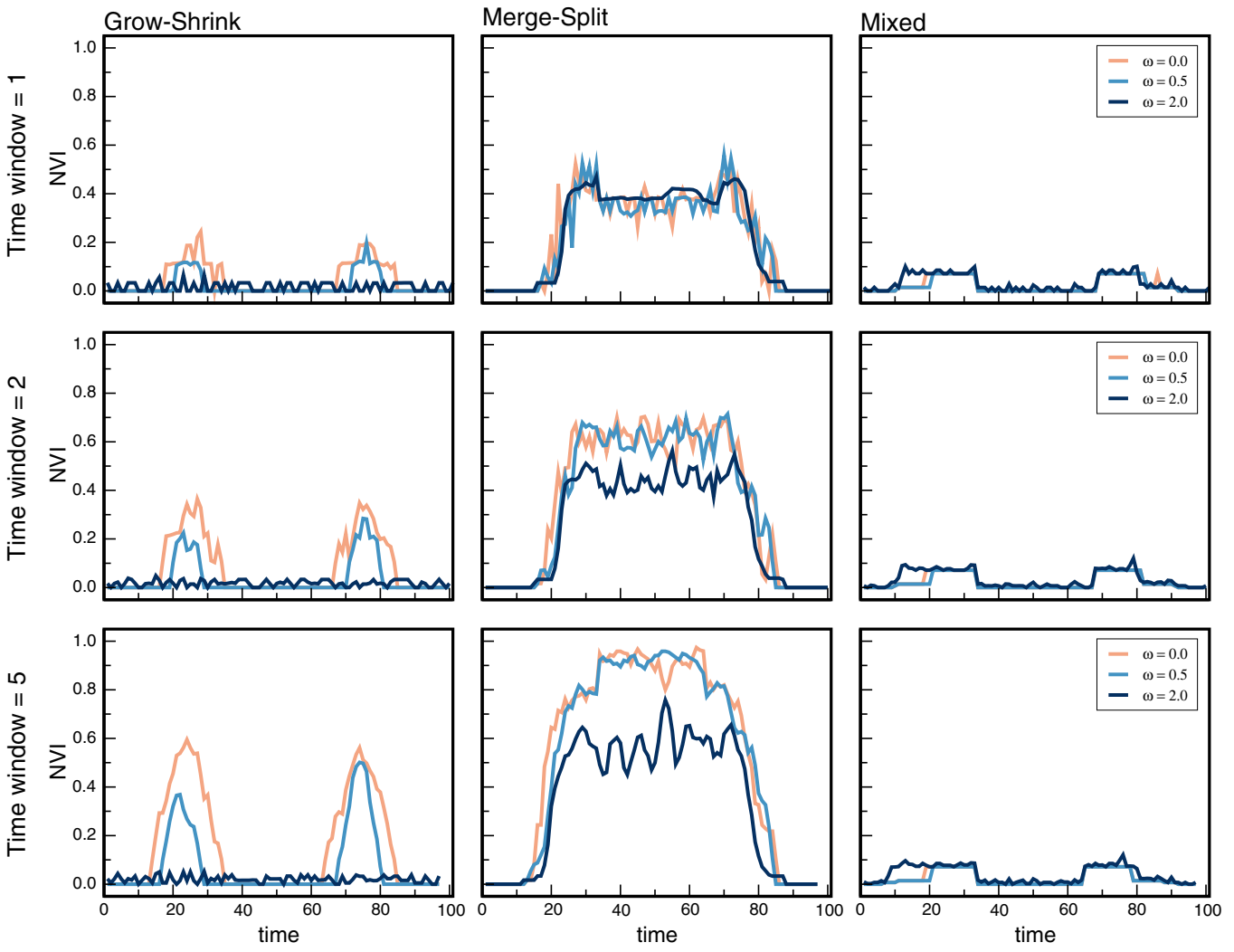

FIG. 4. (Color online) Plots of the normalized variation of information (NVI) between the planted partition and the results of the multislice algorithm in Fig. 3, for three different interslice couplings and for the three benchmarks proposed. The NVI is computed using the proposed evolving formulation and for three different window sizes: 1, 2, and 5. There is a column for each benchmark and a row for each time window size.

Figure 4 shows the results for the NVI. We observe that, for the grow-shrink benchmark, the error is large for $\omega = 0$, but becomes almost zero at $\omega = 2$. Moreover, the values of the NVI increase with the size of the time window for $\omega = 0$ and $\omega = 0.5$, but in a larger amount when the parameter corresponds to the static version of the multislice algorithm. This means that the interslice weight is helping to find the persistence of nodes in their communities, as expected. The merge-split benchmark shows an almost identical bad performance for the three values of ω at windows of size 1, but $\omega = 2$ does not make it worse when the size of the window increases, unlike the other two. The mixed benchmark is quite neutral, with just a small difference from $\omega = 2$. Finally, the NVI squared errors reported in Table I and calculated using Eq. (15) are in perfect agreement with this analysis. The results using the NMI and Jaccard indices (see Ref. [29]) also support these observations. Thus, we may conclude that, in this case, the use of memory to track the evolution of communities is convenient, but the trade-off between the continuity of the community structure and its static relevance must be carefully adjusted.

TABLE I. The NVI squared error, for each method tested and each benchmark in Fig. 3, considering three different time windows.

	Time	NVI squared error		
Multislice	window	Grow-shrink	Merge-split	Mixed
	1	0.0065	0.0851	0.0015
$\omega = 0.0$	2	0.0201	0.2146	0.0015
	5	0.0658	0.4427	0.0016
	1	0.0023	0.0808	0.0014
$\omega = 0.5$	2	0.0067	0.2019	0.0014
	5	0.0242	0.4278	0.0015
	1	0.0006	0.0878	0.0023
$\omega = 2.0$	2	0.0005	0.1113	0.0024
	5	0.0006	0.1922	0.0029

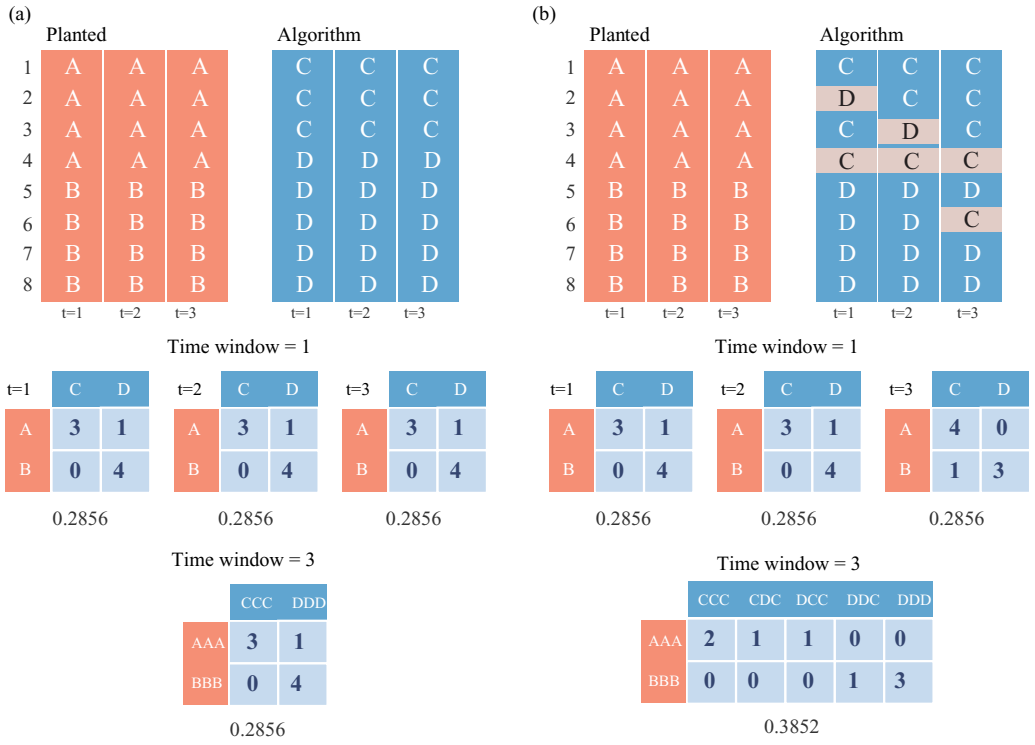

FIG. 5. (Color online) Example of the comparison of a planted evolving community structure with the results from two different algorithms. The NVI values are shown below the corresponding contingency tables for time windows of sizes 1 and 3.

V. CONCLUSION

We have presented a simple model based on the stochastic block model that allows for the construction of time-dependent networks with evolving community structure. It is useful for benchmarking purposes in testing the ability of community detection algorithms to track properly the structural evolution. We have also introduced extended time-dependent measures for the comparison of different partitions in the dynamic case, which allow for the observation of differences between the outcome of the algorithms and the planted partitions through time. Our code for benchmark generation and the time-dependent comparison indices is available at [33] and released under the GNU General Public License.

ACKNOWLEDGMENTS

This work was partially supported by MINECO through Grant No. FIS2012-38266 and by the EC FET-Proactive Project PLEXMATH (Grant No. 317614). A.A. also acknowledges partial financial support from the ICREA Academia and the James S. McDonnell Foundation. R.K.D. and S.F. gratefully acknowledge EC FET-Proactive Project MULTIPLEX, Grant No. 317532.

APPENDIX: DISTINGUISHING COMMUNITY EVOLUTIONS WITH WINDOWED MEASURES

Figure 5 shows an example in which, according to the planted partitions, the eight nodes of a network are divided in two communities of four nodes each and these partitions remain constant throughout the three times steps of the network evolution. Two different community detection algorithms find the communities evolutions represented in Figs. 5(a) and 5(b), which are characterized by the assignment of just one node to the wrong community at each time step. In Fig. 5(a) this node is the fourth one during the three time steps, while in Fig. 5(b) they are the second, the third, and the sixth, respectively. Since the nature of the mistake is the same at all time steps, the comparison of the planted and algorithm partitions with a time window of size 1 generates equivalent contingency tables, thus the standard comparison measures do not change in time, with a constant value of the NVI equal to 0.2856. However, if we take into account a time window of size 3, the two evolving community structures detected by the algorithms are different, yielding structurally different contingency tables and values of the NVI equal to 0.2856 and 0.3852, respectively. Therefore, the conclusion is that windowed measures give complementary information for the comparison of time evolving community structures due to their capacity to take into account several snapshots at the same time.

[1] L. Kovanen, M. Karsai, K. Kaski, J. Kértesz, and J. Saramäki, J. Stat. Mech. (2011) P11005.
[2] P. Holme and J. Saramäki, Phys. Rep. **519**, 97 (2012).
[3] N. Perra, A. Baronchelli, D. Mocanu, B. Gonçalves, R. Pastor-Satorras, and A. Vespignani, Phys. Rev. Lett. **109**, 238701 (2012).

[4] M. Starnini, A. Baronchelli, A. Barrat, and R. Pastor-Satorras, Phys. Rev. E **85**, 056115 (2012).
[5] A. Barrat, B. Fernandez, K. K. Lin, and L.-S. Young, Phys. Rev. Lett. **110**, 158702 (2013).
[6] S. Fortunato, Phys. Rep. **486**, 75 (2010).
[7] J. Hopcroft, O. Khan, B. Kulis, and B. Selman, Proc. Natl. Acad. Sci. U.S.A. **101**, 5249 (2004).
[8] D. Chakrabarti, R. Kumar, and A. Tomkins, *KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, New York, 2006), pp. 554–560.
[9] G. Palla, A.-L. Barabási, and T. Vicsek, Nature (London) **446**, 664 (2007).
[10] J. Ferlez, C. Faloutsos, J. Leskovec, D. Mladenic, and M. Grobelnik, *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering* (IEEE Computer Society, Washington, DC, 2008), pp. 1328–1330.
[11] P. Ronhovde and Z. Nussinov, Phys. Rev. E **80**, 016109 (2009).
[12] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J. P. Onnela, Science **328**, 876 (2010).
[13] C. Granell, S. Gómez, and A. Arenas, Chaos **21**, 016102 (2011).
[14] P. Ronhovde, S. Chakrabarty, D. Hu, M. Sahu, K. K. Sahu, K. F. Kelton, N. A. Mauro, and Z. Nussinov, Sci. Rep. **2**, 329 (2012).
[15] D. S. Bassett, M. A. Porter, N. F. Wymbs, S. T. Grafton, J. M. Carlson, and P. J. Mucha, Chaos **23**, 013142 (2013).
[16] P. Bródka, S. Saganowski, and P. Kazienko, Soc. Network Anal. Min. **3**, 1 (2013).
[17] M. De Domenico, A. Lancichinetti, A. Arenas, and M. Rosvall, Phys. Rev. X **5**, 011027 (2015).
[18] P. Holland, K. B. Laskey, and S. Leinhardt, Soc. Networks **5**, 109 (1983).
[19] M. Girvan and M. E. Newman, Proc. Natl. Acad. Sci. U.S.A. **99**, 7821 (2002).
[20] A. Lancichinetti, S. Fortunato, and F. Radicchi, Phys. Rev. E **78**, 046110 (2008).
[21] R. Guimerà and M. Sales-Pardo, Proc. Natl. Acad. Sci. U.S.A. **106**, 22073 (2009).
[22] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová, Phys. Rev. Lett. **107**, 065701 (2011).
[23] P. Jaccard, New Phytol. **11**, 37 (1912).
[24] W. M. Rand, J. Am. Stat. Assoc. **66**, 846 (1971).
[25] L. Hubert and P. Arabie, J. Classif. **2**, 193 (1985).
[26] A. Strehl and J. Ghosh, J. Mach. Learn. Res. **3**, 583 (2002).
[27] S. V. Dongen, Ph.D. thesis, Dutch National Research Institute for Mathematics and Computer Science, University of Utrecht, 2000.
[28] M. Meilă, J. Multivar. Anal. **98**, 873 (2007).
[29] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevE.92.012805 for the results using the normalized mutual information and the Jaccard index.
[30] J. Reichardt and S. Bornholdt, Phys. Rev. Lett. **93**, 218701 (2004).
[31] http://netwiki.amath.unc.edu/GenLouvain/GenLouvain
[32] G. Petri and P. Expert, Phys. Rev. E **90**, 022813 (2014).
[33] http://rkd.zgib.net/proj/multiplex/

Supplemental material

A benchmark model to assess community structure in evolving networks

C. Granell, R. K. Darst, A. Arenas, S. Fortunato, S. Gómez

	Time window	Jaccard squared error Grow/Shrink	Merge/Split	Mixed
Multislice $\omega = 0.0$	1	0.0720	0.3345	0.0307
	2	0.1365	0.4840	0.0303
	5	0.2336	0.6272	0.0325
Multislice $\omega = 0.5$	1	0.0293	0.3193	0.0276
	2	0.0546	0.4608	0.0282
	5	0.1105	0.6013	0.0303
Multislice $\omega = 2.0$	1	0.0019	0.3326	0.0360
	2	0.0014	0.3605	0.0374
	5	0.0147	0.4488	0.0421

Table S1: Jaccard squared error, for each method tested and each benchmark, considering three different time windows.

	Time window	NMI squared error Grow/Shrink	Merge/Split	Mixed
Multislice $\omega = 0.0$	1	0.0337	0.4932	0.0067
	2	0.0621	0.4806	0.0063
	5	0.1022	0.4855	0.0059
Multislice $\omega = 0.5$	1	0.0143	0.4896	0.0060
	2	0.0262	0.4753	0.0059
	5	0.0479	0.4790	0.0055
Multislice $\omega = 2.0$	1	0.0065	0.4951	0.0094
	2	0.0041	0.4891	0.0094
	5	0.0041	0.4825	0.0100

Table S2: NMI squared error, for each method tested and each benchmark, considering three different time windows.

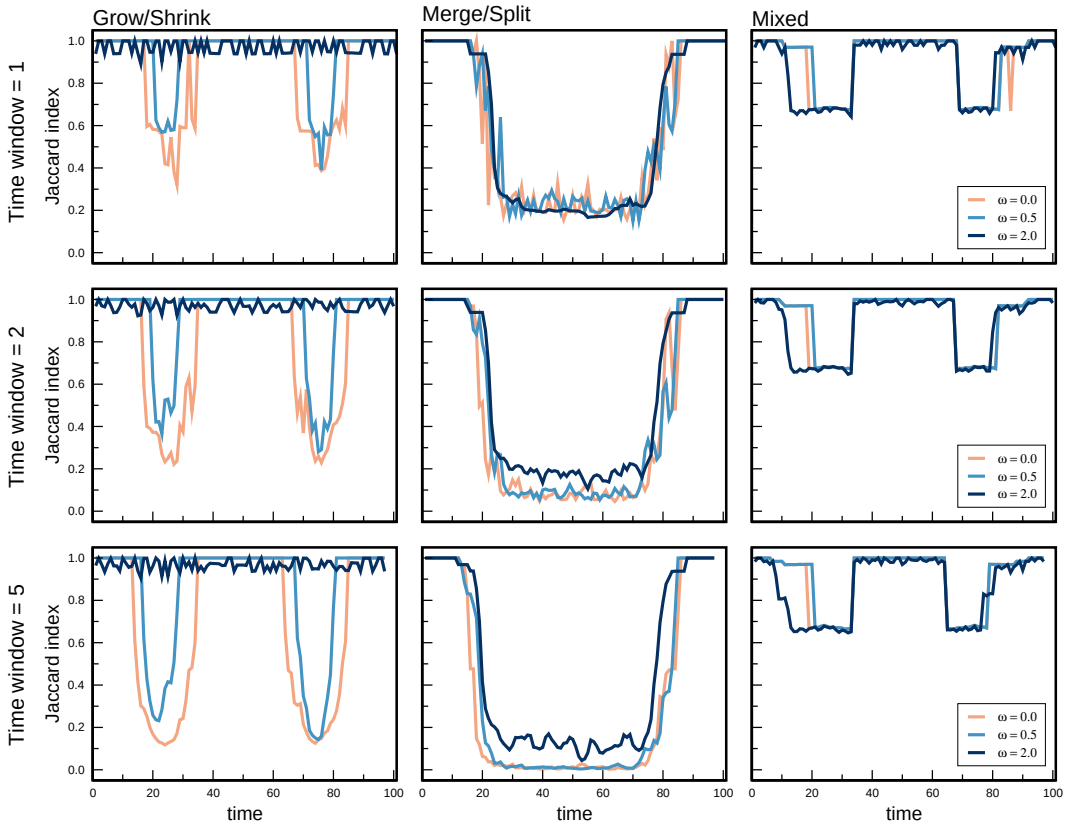

Figure S1: Plots of the Jaccard Index between the planted partition and the results of the multislice algorithm for three different inter-slice couplings and for the three benchmarks proposed. The Jaccard index is computed using the proposed evolving formulation and for three different window sizes: 1, 2 and 5. There is a column for each benchmark, and a row for each time window size.

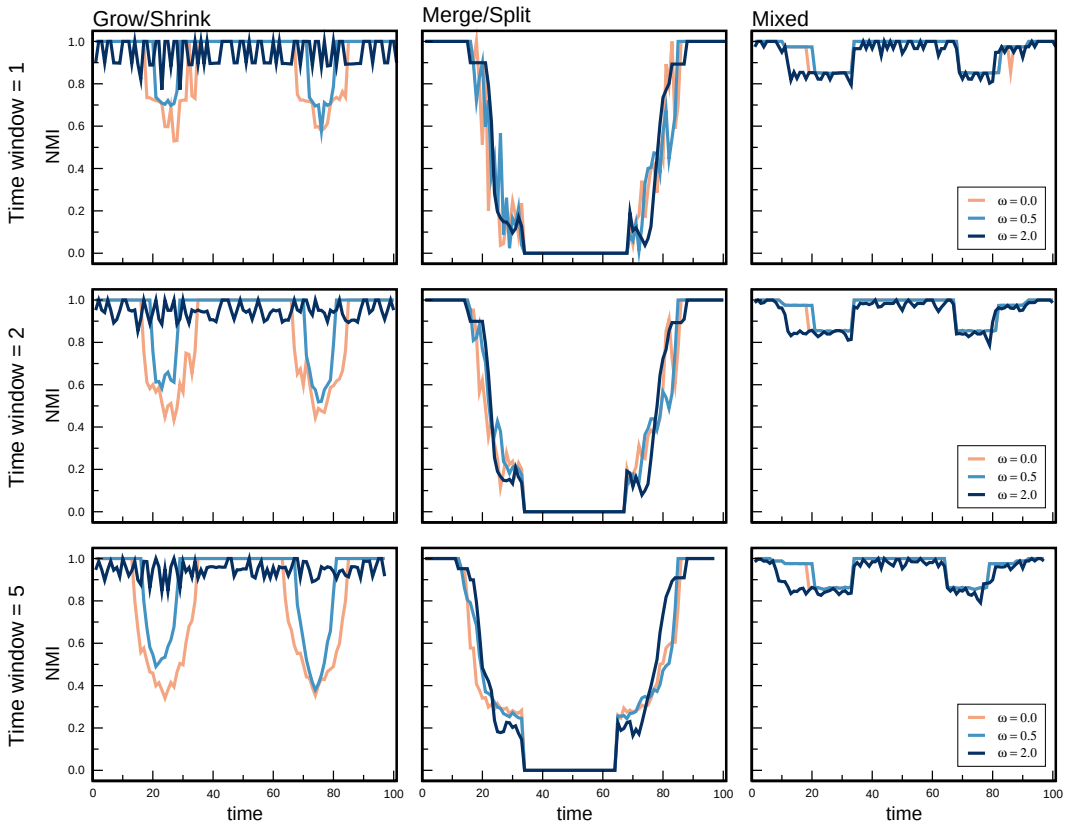

Figure S2: Plots of the Normalized Mutual Information (NMI) between the planted partition and the results of the multislice algorithm for three different inter-slice couplings and for the three benchmarks proposed. The NMI is computed using the proposed evolving formulation and for three different window sizes: 1, 2 and 5. There is a column for each benchmark, and a row for each time window size.