A benchmark model to assess community structure in evolving networks

Clara Granell, Richard K. Darst, Alex Arenas, Santo Fortunato, and Sergio Gómez

1Departament d’Enginyeria Informàtica i Matemàtiques, Universitat Rovira i Virgili, 43007 Tarragona, Spain
2Complex Systems Unit, Aalto University School of Science, P.O. Box 12200, 00076, Finland
3IPHES, Institució Catalana de Paleoeconomia Humana i Evolució Social, 43007 Tarragona, Spain

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 afterwards 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 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. Additionally, we perform several tests on various algorithms which show, unsurprisingly, that dynamic techniques are more suitable than static ones to describe community evolution.

PACS numbers: 89.75.Fb, 89.75.Hc

1. INTRODUCTION

The analysis and modeling of temporal networks has received a lot of attention lately, mainly due to the increasing availability of time-stamped network datasets. A relevant issue is whether and how the community structure of networks 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, e.g. rapid growth or disruption.

Indeed, there has been a lot of activity around this topic over the past years. 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 does it mean 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/deletion of nodes/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 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, that generates three classes of dynamic benchmark graphs. The objective is to provide with 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.

The paper is structured as follows. In Section 2 we describe the model to generate the benchmark networks, Section 3 introduces measures of comparison between dynamic clusterings, Section 4 shows the application of several static and dynamic algorithms on the proposed benchmarks, and Section 5 reports our conclusions. Additional plots that are omitted in the main text for spatial reasons are offered in the Appendix.

2. MODEL DESCRIPTION

The model for generating networks with evolving community structure we propose is based on the classic stochastic block model (SBM). It works as follows: a network is divided in 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. If $p_{in} > p_{out}$ the resulting subgraphs are communities, as the (expected) link density within subgraphs exceeds the link density between subgraphs. The generation of samples from this model has a built-in efficiency: If there are
\( m_{\text{max}} \) pairs of nodes, the actual number of edges is drawn from a binomial distribution with parameters \( m_{\text{max}} \) 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/shrinking and merging/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 which 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 intra-community link density \( p_{\text{in}} \) and inter-community link density \( p_{\text{out}} \). 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, that 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 \( n_2 = 2n - n_1 \). Then, at time \( t \) the number of nodes in community A is

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

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

\[
x(t) = \begin{cases} 
2t^* & 0 \leq t^* < 1/2 \\
2 - 2t^* & 1/2 \leq t^* < 1
\end{cases}
\tag{2}
\]

(with \( t^* \equiv (t/\tau + \phi) \mod 1 \)) which controls the time periodicity. The constant \( \phi \) 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 \( t/\tau \mod 1 = 0, 1/4, 2/4, \) and \( 3/4 \) 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 of \( p_{\text{in}} \) 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 \( p_{\text{in}} \) and between \( j \) and each node in community A with equal and independent probability \( p_{\text{out}} \).

thus the ensemble is maintained. Conveniently, all edges can be pre-computed 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.

A special case which we need to cope with is the situation where \( f \) is very high and \( p_{\text{in}} \) 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 re-run.

Fig. 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 \( p_{\text{in}} \), where those links are placed at initialization and kept unmodified over time. The extreme states are two: the unmerged and the merged state. In the unmerged state, all possible pairs of nodes between the two communities

\[
\begin{align*}
\text{(a) Grow / Shrink} & \\
\text{(b) Split / Merge} & \\
\text{(c) Mixed} &
\end{align*}
\]
have an edge with probability \( p_\text{out} \). This means the network still has a connected component, but the the nodes form two communities. In the merged state, all possible pairs of nodes between these two communities have an edge with probability \( p_\text{in} \), which implies that all pairs of nodes in the network have the same link density \( p_\text{in} \), 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 inter-community edges in the unmerged state, \( m_\text{um} \), and in the merged state, \( m_\text{m} \), are first picked from a binomial distribution consistent with the binomial distribution parameters \( n^2 \) and \( p_\text{out} \) or \( p_\text{in} \). All possible inter-community edges are placed in some arbitrary but random order, and the first

\[
m^*(t) = (1 - x(t))m_\text{um} + x(t)m_\text{m}
\]

edges are selected to be active at time \( t \). The effective intra-community link density is \( p_\text{in}^* = m^*(t)/n^2 \). The parameter \( x(t) \) is the triangular waveform from Eq. 2. In practice, this means 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 inter-community link density is \( p_\text{in}^* = p_\text{in} \) (at \( t = \tau/2 \)). However, due to the detectability limit of communities in stochastic block models, this is not the case [18]. Even when \( p_\text{out} < p_\text{in} \), it can be that the configuration is indistinguishable from one large community. Following [18], at the point

\[
p_\text{in} - p_\text{in}^* = \sqrt{\frac{1}{n_W}(p_\text{in} + p_\text{in}^*)}
\]

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 are a total of \( 4n \) nodes with two merging/splitting communities \( (2n \) nodes) and two growing/shrinking communities \( (2n \) nodes). The intra-community links are managed with the same processes as above with phase factors \( \phi = 0 \) 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 \( p_\text{out} \). Fig. 1(c) exemplifies the mixed benchmark where \( q = 4 \).

### 3. Time-Dependent Comparison Indices

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 [19], the Rand index [20], the adjusted Rand index [21], the normalized mutual information [22], the van Dongen metric [23] and the normalized variation of information metric [24]. 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 \( C = \{C_\alpha | \alpha = 1, \ldots, r \} \) and \( C' = \{C'_{\alpha'} | \alpha' = 1, \ldots, r' \} \) 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 \( C_\alpha \) and \( C'_{\alpha'} \),

\[
m_{\alpha\alpha'} = |C_\alpha \cap C'_{\alpha'}|, \tag{5}
\]

The sizes of the clusters simply read as \( n_\alpha = |C_\alpha| \) and \( n'_{\alpha'} = |C'_{\alpha'}| \). Then for each snapshot, see Fig. 2(a). How-

da, the Jaccard index,

\[
J = \frac{\sum_{\alpha, \alpha'} m_{\alpha\alpha'}^2}{\sum_{\alpha, \alpha'} (m_{\alpha\alpha'}^2 + m'_{\alpha\alpha'})}, \tag{6}
\]

the normalized mutual information index,

\[
\text{NMI} = \frac{-2 \sum_{\alpha, \alpha'} m_{\alpha\alpha'} \log \frac{N m_{\alpha\alpha'}}{n_\alpha n'_{\alpha'}}}{\sum_{\alpha} n_\alpha \log \frac{n_\alpha}{N} + \sum_{\alpha'} n'_{\alpha'} \log \frac{n'_{\alpha'}}{N}}, \tag{7}
\]

and the normalized variation of information metric,

\[
\text{NVI} = \frac{-1}{\log N} \sum_{\alpha, \alpha'} \frac{m_{\alpha\alpha'}^2}{n_\alpha n'_{\alpha'}} \log \frac{m_{\alpha\alpha'}^2}{n_\alpha n'_{\alpha'}}, \tag{8}
\]

where, by convention, \( \log 0 = 0 \).

In the case of evolving networks we have to compare two sequences of partitions, \( \{C(t)| t = 1, \ldots, T \} \) and \( \{C'(t)| t = 1, \ldots, T \} \), a task which 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 \( C(t) \) and \( C'(t) \) 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 indexes and metrics, obtained by considering sequences of consecutive partitions, i.e. time windows of a predefined duration \( \tau \). In Fig. 2(b) we show the comparison between individual snapshots and sequences.
which accounts for the number of nodes following the same cluster evolutions $D_\alpha(t; \tau)$ and $D'_\alpha(t; \tau)$. Likewise, we have

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

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

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

Finally, we may use Eqs. (6) to (8) to calculate the corresponding windowed Jaccard index $J(t; \tau)$, windowed normalized mutual information index $\text{NMI}(t; \tau)$ and windowed normalized variation of information metric $\text{NVI}(t; \tau)$, respectively. Of course, the windowed measures reduce to the standard static ones when $\tau = 0$.

4. RESULTS

Here we analyze the performance of different community detection algorithms when asked to reveal the community structure of our benchmarks. We put to test three widely used static community detection algorithms: modularity optimization [25], the Louvain algorithm [26] and Infomap [27]. Modularity is a measure which rates the quality of a particular partitioning of a network into communities, where high modularity indicates that the fraction of links internal to the communities is greater to what would be expected in a random graph with the same degree distribution. Modularity is a versatile tool, and it can be used in networks which are even directed and signed [28]. In this work, the modularity optimization is done using a combination of heuristics consisting of tabu search [29], fast algorithm [30] and the reposition algorithm, an adaptation of the Kernighan-Lin method [31]. The Louvain algorithm is also based on modularity, and it performs repeatedly a two-step process that consists in a local optimization and a posterior coarse-graining of the communities into single nodes. The updated code from [32] is used, and the method is invoked 10 times, taking the run which maximizes modularity (of the most granular level) for calculations. Lastly, we use the Infomap algorithm, a method that decomposes a network into modules by optimally compressing a description of information flows on the network. All three algorithms were designed to be applied on single networks, thus to use them to unveil the community structure of the evolving benchmarks we must treat each layer (snapshot) independently from the previous one.

On the other hand, the analysis of communities using the information about the evolution of the network is done using the extension of modularity to multi-layer networks published in [11]. In their representation, each layer consists of a single network at a particular time. The layers are connected between them by joining each node with its counterpart in the next and previous layer, and this link has a specified weight $\omega$, equal for all links of this kind, which acts as a tuning parameter. For $\omega = 0$, no connection between layers 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 $\gamma$, which accounts for the tuning of the resolution at which communities are found, a la [\ref{33}]. In this work, we have used the code available in [\ref{34}], setting the resolution parameter $\gamma$ to one and varying the inter-slice coupling $\omega$.

The benchmarks used to put to test these algorithms 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 of with $q = 2$ communities, where each community has initially $n = 32$ nodes (therefore the total size of the network is $N = 64$), with $p_{\text{in}} = 0.5$, $p_{\text{out}} = 0$, and $f = 0.5$, $\tau = 100$ 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 $p_{\text{in}} = 0.5$, $p_{\text{out}} = 0$, 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. With these parameters, the benchmark is trivially simple. We do this in order to show that, even in this case, history can be important for an accurate detection. Future work can vary $p_{\text{in}}$ and $p_{\text{out}}$ in order to determine the relative accuracies of the methods under increasing confusion (smaller $p_{\text{in}} - p_{\text{out}}$).

Figure [\ref{fig:results}] illustrates the results of applying the three static algorithms to the three proposed benchmarks. To quantitatively evaluate the accuracy of the algorithms tested, we use the measures introduced in the previous chapter. We calculate the measures between the partitions obtained by the algorithms and the planted ones, for two values of the time window. When the time window is one ($\tau = 0$), each snapshot is considered independently, while with time window = 2 ($\tau = 1$), the evolution of the partitions during two time steps is also evaluated. For the sake of clarity, Fig. [\ref{fig:results}] shows the results for the Normalized Mutual Information only, and the NVI and the Jaccard Index are shown in the Appendix. As we can see, in most cases the three algorithms fail to detect the most abrupt changes in the benchmarks.

For the application of the time evolving community detection algorithm, the inter-slice coupling parameter must be tuned. After performing multiple tests, we chose to represent in Fig. [\ref{fig:results}] the most relevant cases: in the extreme case $\omega = 0$ slices are considered independently, $\omega = 0.8$ is an intermediate value which provides good results, and $\omega = 4$ provides an example of the partitioning obtained when using strong coupling between layers.

We observe in Fig. [\ref{fig:results}] that the accuracy of the community detection in [\ref{11}], when the inter-slice coupling is different from zero, is slightly better than the accuracy obtained in any of the static methods evaluated, indicating the convenience of using memory in tracking the evolution of communities, as expected.

5. CONCLUSIONS

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 [\ref{35}] and released under the GNU General Public License.

Acknowledgments

This work has been partially supported by MINECO through Grant FIS2012-38266; and by the EC FET-Proactive Project PLEXMATH (grant 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 MULTIPLEX, grant number 317532 of the European Commission and the computational resources provided by Aalto University Science-IT project.

Appendix

Along with the Normalized Mutual Information, we proposed two additional generalized measures for calculating the similarity between partitions of time-evolving communities (Section 3 of the main text). Those were the Jaccard Index and the Normalized Variation of Information. Fig. [\ref{fig:jaccard}] and Fig. [\ref{fig:nvi}] are the plots for the two additional measures for the tests evaluating the static algorithms, Infomap, Louvain and Modularity optimization. As we can observe, these two measures agree with the previously shown NMI, however the Jaccard Index is more penalizing with the discrepancies between the outcome of the algorithms and the planted partitions to be found.

The two additional measures, Jaccard Index and Variation of Information, for the evolving community detection algorithm are presented in Fig. [\ref{fig:jaccard}] and Fig. [\ref{fig:nvi}] respectively.

References

FIG. 3: Results of the application of three static community detection algorithms 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 each one of the algorithms. 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. Let us make clear to the reader the interpretation of the plots: the grow/shrink benchmark consists of a network of 64 nodes, forming initially two equally sized communities, denoted by different hues of blue. We observe that, as time goes by, the light blue community grows, while the dark blue shrinks, until approximately $t = 25$, when the dark community grows and the light one shrinks. At time $t = 75$ the process is reversed again. Following the same column, one can see that the Infomap algorithm is able to grasp the expected evolving community structure, while the two other algorithms tend to split the largest community in two when the difference between the sizes of the communities is the maximum. The same reasoning applies to the other two benchmarks. We observe that, in general, the three algorithms fail to detect the underlying community structure, which is not surprising, due to their static nature.

FIG. 4: Plots of the Normalized Mutual Information (NMI) between the planted partition and the result of each static algorithm, for the three benchmarks proposed, using the evolving formulation of the NMI for two window sizes. In the first row, we plot the NMI between the planted partition and the algorithm at study considering only one snapshot at a time. That is, we have computed the NMI between the planted partition at $t = 1$ and the algorithm’s result at $t = 1$, and we have repeated this process until $t = T_{\text{max}}$. The second row is calculated taking into account the evolving nature of the benchmark. Here, NMI is calculated between pairs of snapshots. That is, comparing the planted partition at $t = [1, 2]$ with the algorithm’s result at $t = [1, 2]$. 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 is also the same.

[34] URL http://netwiki.amath.unc.edu/GenLouvain/GenLouvain
[35] URL http://rkd.zgib.net/proj/multiplex/
FIG. 5: Results of the application of a dynamic community detection method to the previous benchmarks. Rows denote different values of the inter-slice parameter $\omega$, which is the weight between different instances of the same nodes across layers. When $\omega = 0$ the layers are disconnected and then the community detection analysis is done for each layer separately. As this value increases, more importance is given to the evolving nature of the problem, and communities across layers are found.
FIG. 6: Plot of the NMI between the result of the dynamic algorithm and the planted partitions, for different values of the inter-slice coupling, and for two values of the time window.

FIG. 7: Plot of the Jaccard Index between the partitions obtained with the three static algorithms (indicated with different colors) and the planted partitions of the benchmark. First row corresponds to the static version of the Jaccard Index, and the second takes into account the evolving nature, evaluating two snapshots at a time.
FIG. 8: Plot of the Normalized Variation of Information between the partitions obtained with the three static algorithms (indicated with different colors) and the planted partitions of the benchmark. First row corresponds to the static version of the NVI measure, and the second takes into account the evolving nature, evaluating two snapshots at a time.

FIG. 9: Plot of the Jaccard Index between the partitions obtained with the dynamic community detection method and the planted partitions of the benchmark, for three values of the inter-slice coupling (denoted by different colors). First row corresponds to the static version of the Jaccard Index, and the second takes into account the evolving nature, evaluating two snapshots at a time.
FIG. 10: Plot of the Normalized Variation of Information between the partitions obtained with the dynamic community detection method and the planted partitions of the benchmark, for three values of the inter-slice coupling (denoted by different colors). First row corresponds to the static version of the NVI measure, and the second takes into account the evolving nature, evaluating two snapshots at a time.