Detecting Changes in Sequences of Attributed Graphs

Daniele Zambon*[1], Lorenzo Livi[2], and Cesare Alippi[1 3]

[1]Università della Svizzera italiana, Lugano, Switzerland.
[2]University of Exeter, Exeter, United Kingdom.
[3]Politecnico di Milano, Milano, Italy.

Abstract

In several application domains, one deals with data described by elements and their pair-wise relations. Accordingly, graph theory offers a sound framework to cast modeling and analysis tasks. However, due to a multitude of reasons, real-world systems change operating conditions over time, hence calling for time-dependent, graph-based representations of systems' state. Here, we deal with this problem by considering a methodology for detecting changes in sequences of graphs. The adopted methodology allows to process attributed graphs with variable order and topology, and for which a one-to-one vertex correspondence at different time steps is not given. In practice, changes are recognized by embedding each graph into a vector space, where conventional change detection procedures exist and can be easily applied. Theoretical results are presented in a companion paper. In this paper, we introduce the methodology and focus on expanding experimental evaluations on controlled yet relevant examples involving geometric graphs and Markov chains.

1 Introduction

Research is giving continuously more attention to graphs as a representation tool for complex entities, e.g., networked systems or space and time-related data. We identify two principal directions that study dynamics related to graphs: graph signal processing [1], which considers a graph as the spatial domain for signals, and temporal networks [2], that study the evolution of the graph dynamic itself.

This paper lays in the temporal network framework and considers a network-generating process $\mathcal{P}$ producing, over time, sequences of graphs. In the following, we will consider *attributed graphs* as it is a very general family of graphs that, e.g., includes directed and undirected labelled graphs characterized by a variable number of vertices and edges [3]. The type of possible labels associated to nodes and/or edges is rich, e.g., one might have numerical, categorical, and hybrid labels. In addition, multiple attributes can be associated to the same vertex/edge. The stochastic process $\mathcal{P}$ generates a stream of graphs $g_1, g_2, \dots, g_t, \dots$ contained in the domain $\mathcal{G}$. In monitoring the process $\mathcal{P}$, a relevant problem is the detection of changes in stationarity, e.g., induced by long lasting anomalies, occurring events or a source of concept drifts. Within the stationarity hypothesis the process generates graphs according to a specific, *nominal*, probability distribution Q. Detecting a change in stationarity requires detecting the unknown time τ from which $\mathcal{P}$ starts generating graphs according to a *non-nominal* distribution

*Corresponding author, `daniele.zambon@usi.ch` .

Published in 2017 IEEE Symposium Series on Computational Intelligence (SSCI).

DOI: 10.1109/SSCI.2017.8285273.

©2017 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

$\widetilde{Q} \neq Q$, i.e.,

$$g_t \sim \begin{cases} Q & t < \tau, \\ \widetilde{Q} & t \geq \tau. \end{cases}$$

The time τ is said to be the change point.

Detection in numeric time series is usually addressed via either change point methods [4] or change detection tests [5]. The former takes into account time series in an off-line fashion, when the data have already been completely collected; the latter adopts a sequential strategy for processing incoming observations. Originally, the methods assumed normal distributed data. Further developments extended these techniques for multivariate datastreams with non-Gaussian underlying distributions.

Considering graph streams, only few works tackle the change detection problem in a classical change detection framework [6, 7]. In particular, an overview of proposed approaches for anomaly and change detection on time-variant graphs is reported in [8], where the authors distinguish the impact of a change on the graph (such as changes involving specific edges/vertices or sub-graphs).

The limitations of these methods, we believe, lay in the difficulty in applying basic mathematical and statistical tools to graphs, e.g., computing the mean of a set of graphs or other similar basic operations. Such issues are especially observed in families of graphs with a variable number of nodes and/or non-conventional attributes, as it happens for attributed graphs. A key problem in analysing such generic objects is the assessment of pairwise dissimilarities [9]. A first fashion of dissimilarity measures analyses graphs in their original domain; a second one consists in mapping graphs to numeric vectors. In this work, we adopt a specific method taken from the Graph Edit Distance (GED) family of graph dissimilarities, whose aim is to create a (partial) matching between the vertices of two graphs and quantify the cost of editing the first graphs in order obtain the second one [10]; the total editing cost is based on the costs of all simple editing operations involved, like vertex/edge insertion/deletion and attribute modification.

The problem formulation as a change detection one tries to be as general as possible to cover all applications modelable as attributed graphs, like abrupt changes, smooth drifts or transient anomalies lasting for a reasonable lapse of time. In this paper, we adopt the methodology first presented in [11], inspect the proposed change detection method more closely, and expand the experiments by addressing specific and controlled situations to test and assess the performances of the method. The suitably generated datasets of variable complexity simulate streams of geometric and statistical graphs with fixed number of vertices but varying topology and attributes.

The remainder of this paper is structured as follows. Section 2 introduces the method for change detection on streams of attributed graphs. Sections 2.0.1 and 2.0.2 describe the two main stages on which the method is designed. Section 3 reports the experimental results that we obtained by studying synthetic streams of graphs. Finally, Section 4 offers concluding remarks.

2 Change detection in a stream of graphs

The method for change detection on a stream of graphs operates in two stages. Given a stream $g_1, g_2, \dots, g_t, \dots$ of labelled graphs we map (embed) the generic graph g_t into a vector $y_t \in \mathbb{R}^M$. As a specific choice of embedding we consider the Dissimilarity Representation [12], which we describe in the next section. Secondly, in order to identify a change in stationarity in $\mathcal{P}$, we monitor the generated multivariate vector stream $y_1, y_2, \dots, y_t, \dots$ and perform a change detection test on it. The two phases are discussed in the sequel; theoretical results regarding the soundness of the current method are detailed in [11].

2.0.1 Dissimilarity representation

The dissimilarity representation operates from a generic set of objects $\mathcal{G}$ (here graphs), and associates to each object a vector in $\mathcal{D} = \mathbb{R}^M$ through a map $\zeta : \mathcal{G} \to \mathcal{D}$. The embedding map is determined by a prototype set $R = \{r_1, \dots, r_M\} \subset \mathcal{G}$ and a distance measure $d(\cdot,\cdot) : \mathcal{G} \times \mathcal{G} \to \mathbb{R}_+$ on the graph domain. This technique associates to generic graphs $g_t \in \mathcal{G}$ vectors $y_t = \zeta(g_t) \in \mathcal{D}$ whose components store the dissimilarities $d(g_t, r_i)$ between the considered graph and the prototypes

$$y_t = \zeta(g_t) := [d(g_t, r_1), \dots, d(g_t, r_M)]^\top .$$

The vector y_t is referred to as the *dissimilarity representation* of g_t. For further details we refer the reader to [12].

The crucial roles of set R and distance $d(\cdot,\cdot)$ determine the effectiveness of the embedding. As mentioned in the introduction, the GED evaluates the cost of editing the first input graph for obtaining the second one. In doing so, it considers both the graph topologies and the discrepancy between matched attributes. Moreover, many parameters and costs can be weighted and tuned to serve the application at hand. As such, we believe this choice is sufficiently versatile.

Regarding the prototype selection technique, we employ the k-Centers method [13]. The method selects prototypes so as to cover training data with balls of equal radius. Specifically, considering a training set $T \subseteq \mathcal{G}$ and starting from an initial guess $R := \{r_1, \dots, r_M\} \subseteq T$, the algorithm operates as follows:

i) for each $r_m \in R$, consider the set C_m of all $g \in T$ such that $d(r_m, g) = \min_{r \in R} d(r, g)$;

ii) for $m = 1, \dots, M$ update the representative r_m with a minimiser $c \in T$ of $\max_{g \in C_m} d(c, g)$;

iii) if the representative set R did not change from the previous iteration then exit, otherwise go to step i).

In conclusion, we compute the prototype set R during the training phase, then, every time process $\mathcal{P}$ produces a new graph g_t, we compute its dissimilarity representation y_t w.r.t. the predefined set R. We obtain a M-dimensional vector stream $y_1, y_2, \dots, y_t, \dots$ on which we apply the change detection test detailed in the next section.

2.0.2 Change detection test

Once the input graphs have been embedded in $\mathbb{R}^M$, the procedure focuses on the y-stream and attains a change detection test suitable for treating multivariate data.

We adopt a variation of the CUSUM change detection test that processes windows $\boldsymbol{y}_w := \{y_t\}_{t=(w-1)n+1}^{w\,n}$ of size n; hence, w defines a numbering system for the windows. For each window w, the test computes a non-negative statistic S_w of $\boldsymbol{y}_w$ and formalises the hypothesis test as:

$$\begin{array}{ll} H_0: & \mathbb{E}[S_w] = 0 \\ H_1: & \mathbb{E}[S_w] > 0. \end{array}$$

S_w has expectation $\mathbb{E}[S_w]$ equal to 0 as far as process $\mathcal{P}$ follows the nominal and stationary condition. Once a change occurs, S_w assumes larger values. Related to it, we consider confidence regions $[0, h_w]$ so that if

$$S_w > h_w \quad \Rightarrow \quad \text{a change is detected} \tag{1}$$

with a certain significance level α; the estimated change time is

$$\widehat{\tau} = \inf\{\, n\, w \,:\, S_w > h_w \,\}.$$

To obtain the statistics S_w, we compare the sample mean $\overline{\boldsymbol{y}}_0$ computed in a training set T with that computed in the w-th window $\boldsymbol{y}_w$, i.e., $\overline{\boldsymbol{y}}_w$. The comparison is attained by the Mahalanobis distance

$$s_w := d_\Sigma(\overline{\boldsymbol{y}}_0, \overline{\boldsymbol{y}}_w) = \sqrt{(\overline{\boldsymbol{y}}_0 - \overline{\boldsymbol{y}}_w)^\top \Sigma^{-1} (\overline{\boldsymbol{y}}_0 - \overline{\boldsymbol{y}}_w)}$$

w.r.t. to a non-singular matrix Σ.

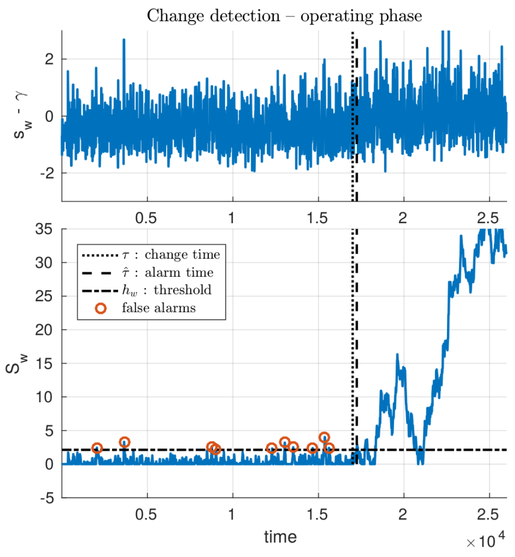

Figure 1: Here is illustrated a sample experiment for showing the characteristic behaviour of the statistics involved in the change detection test, namely Mahalanobis distance s_w, cumulative statistic S_w, threshold h_w, and estimated change point $\hat{\tau}$. In particular, the experiment considers a stream of graphs, and reports the observed statistics s_w and S_w as functions of the window index w. In the top panel, it is possible to notice that statistic $s_w - \gamma$ has continuous fluctuations and it is not easy to spot the change point, at a first glance. Conversely, in the bottom panel, statistic S_w is capable to grasp even subtle variations and enhance them above the threshold h_w. The dashed line represents the window at which the change is detected and the red circles highlight some false alarm in the detection. Once an alarm is raised, statistic S_w immediately get back below threshold h_w during the nominal regime, whereas statistic S_w tends to always increase in the non-nominal regime.

The final statistic S_w inspired by the original CUSUM test is defined as

$$\begin{cases} S_w = \max \{\, 0\,,\ S_{w-1} + (s_w - q)\, \} \\ S_0 = 0, \end{cases} \tag{2}$$

where parameter q can be used to tune the sensitivity of the test. An illustrative experiment has been considered in Figure 1 for showing how statistic S_w cumulates the values of statistic s_w for the detection of a change.

Assuming that $y_1, y_2, \dots, y_t, \dots$ are independent and identically distributed, and that $|T|$ and n are sufficiently large, we consider $\overline{\boldsymbol{y}}_0$ and $\overline{\boldsymbol{y}}_w$ to be Gaussian distributed thanks to the central limit theorem. Under the nominal regime (null hypothesis) they share the same expectation; as a consequence, $\overline{\boldsymbol{y}}_0 - \overline{\boldsymbol{y}}_w$ is normally distributed as well, and it has zero mean. If we consider the sampling variance-covariance matrix of $\overline{\boldsymbol{y}}_0 - \overline{\boldsymbol{y}}_w$ as Σ, for each stationary window w the statistic s_w^2 is distributed as a $\chi^2(M)$.

The threshold h_w in (1) should be set to yield the desired trade-off between specificity and false alarm rate. Since we do not assume a priori knowledge on the distributions $Q, \widetilde{Q}$, a common criterion is to require a specific false alarm rate α, or similarly, a specific *average run length* under null hypothesis of nominal regime (ARL$_0$); in fact, ARL$_0$ is connected to the false alarm rate in the sense that setting a false alarm rate to α at each time steps yields an ARL$_0$ of $\frac{1}{\alpha}$. We numerically estimate the thresholds h_w with the quantiles of order $1-\alpha$ of S_w, as described in Section 3.1.2.

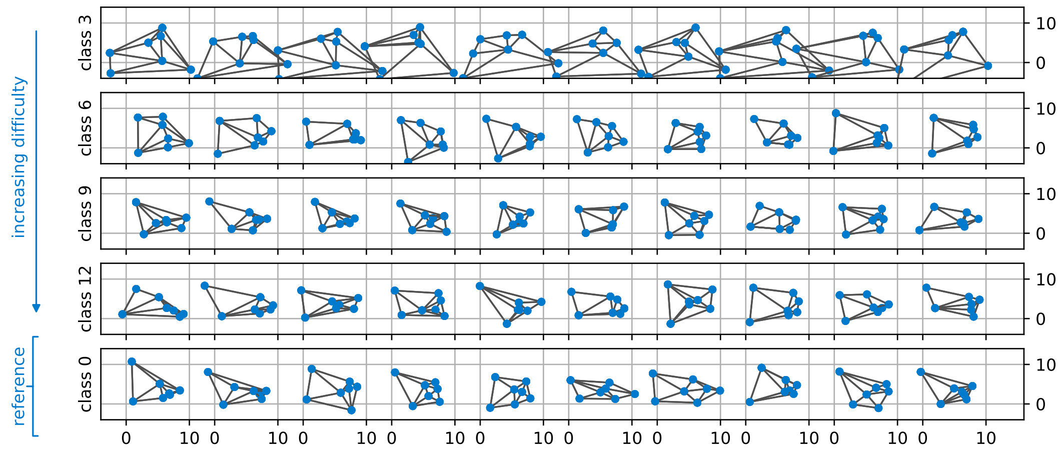

Figure 2: Visual representation of graphs present in the Delunay database. In each row, random instances from the same different class are drawn. The bottom row represents the reference class 0. The other rows show classes 3, 6, 9 and 12. Going from top to bottom, the perturbation that generated the class decreases (see Section 3.1.1). Class 0 is generated by initial points in the square $[0, 10)^2$. It is possible to notice that the Gaussian perturbation sometimes moves the vertices outside the square. Class 3 contains graphs whose vertices are often outside the square; this spread of the graph vertices is introduced by the large radius that generated the initial points of class 3, making classes 3 and 0 easily separable. On the other hand, spread of classes 6, 9 and 12 is smaller and hence related detection problems are more difficult.

3 Experiments

In this section we analyse two aspects of the considered method: the influence of the number M of prototypes and the size n of the window on the detection performance. In the following experiments, we consider synthetic 2-class datasets and generate streams of graphs from those as explained below. Source code for replicating the experiments is available here [14].

3.1 Experiment description

3.1.1 Data

We generate two datasets by considering different mechanisms. Each dataset contains several sub-datasets intended to reproduce increasing levels of difficulty in distinguishing the contained classes, i.e., separation levels.

The first family of dataset is composed of Delaunay triangulations, which are geometric undirected graphs. A first class, class 0, is characterised by seven original points $\{p_i^{(0)}\}_{i=1}^7$, which are drawn from a uniform distribution in the square $[0, 10)^2$. A generic graph of class 0 is defined by a perturbation of the original points. For each component $[p_i^{(0)}]_j$ of original point $p_i^{(0)}$ we drew a nuisance term from a Gaussian distribution $\mathcal{N}(0, 1)$, and added it to $[p_i^{(0)}]_j$; this leads to a set of seven perturbed points. Finally, the Delaunay triangulation computed on this latter set of points constitute the topology of the new graph, and the 2-dimensional position of the points are the vertex attributes.

For obtaining classes with controlled separation levels, we generate the original points $\{p_i^{(c)}\}_{i=1}^7$ of the c-th class, class c, by perturbing the original points $\{p_i^{(0)}\}_{i=1}^7$ of class 0, with a controlled nuisance parameter ρ. Specifically, we added a 2-dimensional vector drawn from a uniform distribution in the circumference of radius

$$\rho_c = 10 \cdot \left(\frac{2}{3}\right)^{c-1} \tag{3}$$

to each $p_i^{(0)}$. Decreasing the radius ρ_c generates more difficult problems in distinguishing class 0 from class c. Following this procedure we generated one thousand graphs per class. Some graph samples are drawn in Figure 2.

The second dataset contains graphs generated from Markov chains with 30 vertices/states and a variable number of edges. There are 5-dimensional vector attributes associated to both vertices and edges. For each sub-dataset, nine hundreds graphs have been created by considering two Markov chains, one for each class. The states of the chain form the vertices and the transition probabilities drive a random walk generating graph topologies. Then, each label is drawn from a 5-variate normal distribution. Therefore, the two classes differ in the transition probabilities (hence the generated topologies) and the expectation of the Gaussian distribution. More details can be found in [15].

With D-c, we refer to the Delaunay sub-dataset composed of class 0 and class c. Analogously, M-c represents classes 0 and 1 in the c-th sub-dataset derived from Markov chains.

Finally, we simulated a stream $g_1, g_2, \dots, g_t, \dots$ by bootstrapping graphs from a class, intended to be the class characterising the nominal distribution of process $\mathcal{P}$. Once reached a predefined change point τ, we bootstrapped from the second class, which defines the non-nominal regime.

3.1.2 Parameters setting

For computing the distance $d(\cdot,\cdot)$, we adopted the bipartite GED implemented [16] using the Volgenant and Jonker assignment algorithm. For matching the real-valued vector attributes, we considered the ℓ^2-distance between them.

Threshold h_w is learned through Monte Carlo. We simulated one million processes of i.i.d. $\chi^2(M)$ observations. Considering the square root of these observations, we emulated the streams $s_1, s_2, \dots, s_w, \dots$ and applied (2) iteratively setting offset q to the square-rooted third quartile of the $\chi^2(M)$ distribution. Finally, we set $\mathrm{ARL}_0 = 100$ and consider the estimated quantile of order $1/\mathrm{ARL}_0$ of S_w as threshold h_w.

The training set is composed of 300+1000 graphs, respectively employed for the prototype selection and the training of the change detection test. Afterwards, the graph stream is produced by the nominal distribution for $8 \cdot n \cdot \mathrm{ARL}_0$ time steps, and $4 \cdot n \cdot \mathrm{ARL}_0$ time steps in the non-nominal regime; we point out here that ARL_0 is with respect to the number of windows.

3.1.3 Figures of merit

In the presence of false alarms, in the non-nominal regime, we might confuse a detected change (true alarm) with an erroneous alarm. In order to assess whether an alarm has been actually triggered by a change or not, we consider a change to be detected whenever the average run length under non-nominal regime (ARL_1) is less than ARL_0.

We repeated every experiment one hundred times and computed the empirical rate of detected changes (DCR) as the ratio of experiments for which the estimated ARL_1 is less then the estimated ARL_0. We estimated the average run length ARL by computing, for each repetition of the same experiment, the mean number of windows between consecutive alarms, then we took the mean of those over all repetitions.

In reporting the results, for assessing statistical significance we considered also the 95% confidence intervals of endpoints the 0.025 and 0.975 quantiles.

3.2 Results

The next sections provide details of performed experiments. As anticipated, the particular choice of prototype set is determinant for an informative representation of the nominal condition. In particular, the number of prototypes determines, in general, the dimension of the embedded space. A first analysis is devoted to compare different number M of prototypes. Another relevant factor is the size n of the windows. On one hand, the larger the window, the more resilient the statistic s_w to outliers in $\boldsymbol{y}_w$. On the other hand, since our knowledge on the distribution of S_w grounds on the central limit theorem, larger windows produce – in principle – more accurate estimates.

3.2.1 Separation level

Figure 3 shows the DCRs, and related 95% confidence intervals, attained by the method in various datasets; indeed, better performance is achieved in easier problems.

In the training phase, we required the thresholds h_w to be such that the ARL_0 would be equal to 100. Table 1 reports the estimated average run lengths ARL_0 and ARL_1. In the Delaunay datasets, there is no statistical evidence for considering the observed ARL_0 different from the target one ($\mathrm{ARL}_0 = 100$). We anticipate that this behaviour is reproduced also in Tables 2, 3, 4 and 5. Conversely, for the Markov datasets, the confidence intervals do not contain the target ARL_0; we believe that this fact might be caused by several factors, e.g., a too short training phase w.r.t. the number of vertices.

For studying the number of prototypes and the window size, we selected two sub-datasets: D-11 and M-2, as they produce sufficiently difficult problems for studying the effects of such parameters.

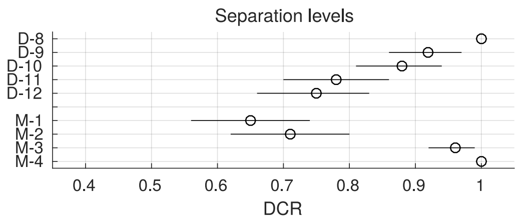

Figure 3: We compare the DCR reached by the method with $M = 4$ prototypes and a window large $n = 32$. We drew a circle for representing the mean DCR obtained across one hundred repetitions and a solid line for representing the 95% confidence interval. We consider both Delaunay and Markov datasets. In D-8, we considered the nominal regime producing graphs from class 0 and non-nominal ones from class 8. Accordingly for the experiments D-9, D-10, D-11, and D-12. Each one of the sub-datasets M-1, M-2, M-3 and M-4 contains two classes, class 0 and class 1, respectively characterising the nominal and non-nominal regimes. We recall that the greater the index of the Delaunay sub-dataset, the more difficult the problem (see (3)). Conversely, more difficult problems in the Markov dataset are identified by smaller index values. The results confirm the expected trends.

3.2.2 Number M of prototypes

By increasing the number of prototypes, in principle, also the useful information retained during the embedding should increase. This does not necessary imply better performance, since in turn, the number of parameters to be estimated increase as M^2 and, accordingly, more training data is required.

Observing both Figures 4 and 5, and starting with the degenerate case with $M = 1$, we notice that the DCR is quite low; in fact, we remark that a DCR not statistically different from 0.5 means that the detection is completely left to the chance. Enlarging the prototype set, we see a general improvement in the detection. A second, and different, criterion for studying the detection is by considering the 95% confidence interval of the observed ARL_1 in Tables 2 and 3. Whenever such interval does not contain

Table 1: ARL estimates related to Figure 3, which are obtained with $M = 4$ prototypes and $n = 32$ window size.

Experiment			ARL_0		ARL_1	
dataset	M	n	mean	95ci	mean	95ci
D-8	4	32	82	[19, 243]	4	[2, 9]
D-9	4	32	82	[19, 243]	17	[4, 67]
D-10	4	32	82	[19, 243]	21	[5, 83]
D-11	4	32	82	[19, 243]	43	[7, 157]
D-12	4	32	82	[19, 243]	41	[8, 148]
M-1	4	32	19	[6, 80]	17	[2, 78]
M-2	4	32	20	[5, 85]	12	[2, 55]
M-3	4	32	16	[6, 77]	4	[2, 17]
M-4	4	32	15	[6, 59]	3	[2, 5]

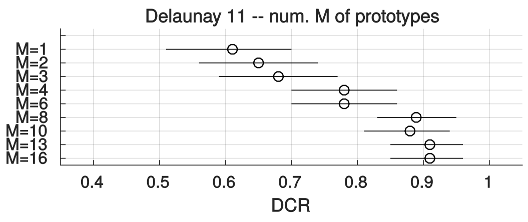

Figure 4: We compare the DCR achieved by the method with a variable number M prototypes and window size fixed to $n = 32$ on problem D-11. Increasing the number of prototypes seems to produce better results, even thought not all estimates are statistically different. A sort of saturation is reached after $M = 8$ prototypes.

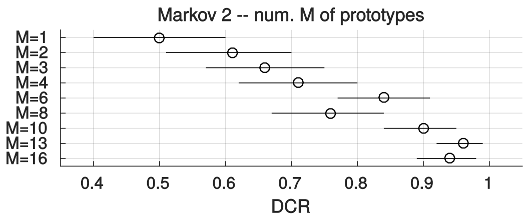

Figure 5: We compare the DCR achieved by the method with variable number M prototypes and window size fixed to $n = 32$ on problem M-2 (Markov chains). With the only exception for $M = 6$ and $M = 8$, increasing the number of prototypes allows to obtain better recognition results, even thought not all estimates are statistically different.

the value ARL_0, we can reject the hypothesis that $\text{ARL}_1 \neq \text{ARL}_0$, hence consider the method capable of detecting all changes.

Part of the results on M-2 have to be considered carefully, in fact some of the ARL_0 estimates are very low w.r.t. the target value 100. This might be symptom of bad parameter settings, e.g., in the GED.

3.2.3 Size n of the windows

Similarly to the study of the number of prototypes in the previous section, we can guess a trend in increasing the window size. In Figure 6, comparing the 95% confidence intervals of DCR, we see that $n = 120$ performs better than setting $n = 36$, and similarly $n = 54$ attains better results than $n = 16$. Consistently, going from short windows to large ones, we identify a decrease in the ARL_1 estimates (Table 4) and an increase of the DCRs (Figure 6), even though in some cases differences are not statistically significant.

Table 2: ARL estimates related to Figure 4, which are obtained with a variable number M prototypes and window size fixed to $n = 32$ on problem D-11.

Experiment			ARL_0		ARL_1	
dataset	M	n	mean	95ci	mean	95ci
D-11	1	32	90	[26, 271]	61	[14, 173]
D-11	2	32	92	[17, 338]	48	[8, 149]
D-11	3	32	79	[20, 272]	42	[6, 150]
D-11	4	32	82	[19, 243]	43	[7, 157]
D-11	6	32	66	[21, 195]	36	[9, 145]
D-11	8	32	62	[18, 136]	25	[7, 84]
D-11	10	32	60	[18, 183]	26	[7, 88]
D-11	13	32	53	[19, 133]	21	[8, 47]
D-11	16	32	40	[16, 102]	19	[7, 45]

Table 3: ARL estimates related to Figure 5, which are obtained on problem M-2 with a variable number M prototypes and window size fixed to $n = 32$.

Experiment			ARL_0		ARL_1	
dataset	M	n	mean	95ci	mean	95ci
M-2	1	32	50	[10, 219]	20	[2, 75]
M-2	2	32	40	[8, 198]	19	[2, 89]
M-2	3	32	20	[6, 109]	18	[2, 122]
M-2	4	32	20	[5, 85]	12	[2, 55]
M-2	6	32	10	[5, 28]	6	[2, 26]
M-2	8	32	7	[4, 15]	8	[2, 45]
M-2	10	32	6	[4, 12]	4	[2, 9]
M-2	13	32	5	[3, 9]	4	[2, 6]
M-2	16	32	5	[3, 8]	3	[2, 5]

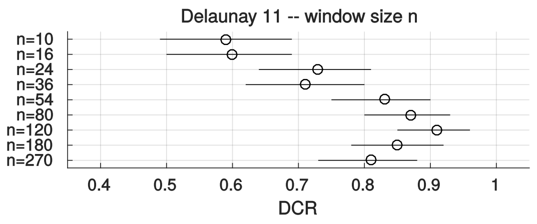

Figure 6: We compare the DCR achieved by the method with variable window size n but fixed number of prototypes $M = 4$ on problem D-11. Enlarging the window seems producing better results. We notice that when the window becomes larger than 120 time steps, performance starts to decrease. We hypothesize that this is due to the need to increase the training size, as the resulting multivariate streams is of higher dimensionality.

Table 4: ARL estimates related to Figure 6, which are obtained with a variable window size n but fixed number of prototypes $M = 4$ on problem D-11.

Experiment			ARL_0		ARL_1	
dataset	M	n	mean	95ci	mean	95ci
D-11	4	10	71	[29, 231]	55	[11, 177]
D-11	4	16	66	[29, 157]	50	[15, 137]
D-11	4	24	77	[26, 196]	45	[12, 146]
D-11	4	36	80	[22, 284]	44	[6, 164]
D-11	4	54	105	[19, 340]	38	[5, 116]
D-11	4	80	97	[13, 368]	27	[4, 113]
D-11	4	120	110	[8, 386]	26	[4, 118]
D-11	4	180	109	[6, 343]	18	[3, 62]
D-11	4	270	80	[6, 319]	16	[2, 86]

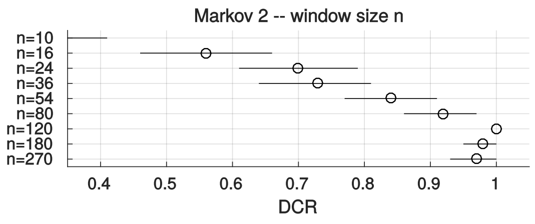

Figure 7: We compare the DCR achieved by the method with variable window size n but fixed number of prototypes $M = 4$ on problem M-2. Enlarging the window produces higher DCRs. The largest windows introduce variability in the estimation but there is no statistical relevance for stating a trend.

Table 5: ARL estimates related to Figure 4, which are obtained with a variable window size n but fixed number of prototypes $M = 4$ on problem M-2.

Experiment			ARL_0		ARL_1	
dataset	M	n	mean	95ci	mean	95ci
M-2	4	10	15	[10, 26]	29	[1, 97]
M-2	4	16	14	[7, 30]	22	[3, 95]
M-2	4	24	17	[6, 76]	15	[2, 61]
M-2	4	36	17	[6, 83]	15	[2, 123]
M-2	4	54	18	[5, 76]	9	[2, 67]
M-2	4	80	23	[6, 148]	6	[2, 31]
M-2	4	120	17	[5, 66]	3	[2, 11]
M-2	4	180	30	[5, 155]	3	[2, 12]
M-2	4	270	35	[4, 281]	3	[2, 6]

Despite Figure 7 reproduces similar behaviour, the location of the DCR confidence interval in M-2, $n = 10$, is unexpectedly shifted towards 0. This fact has not to be considered a valid estimation, since the observed ARL_0 is statistically different from 100, and furthermore is extremely low.

Finally, we remark that, even if visual inspection would suggest that ARL_1 do not increase with the window size n, it does indeed increase; in fact, one might be misled by the fact that the average run lengths are computed w.r.t. the window indexing w, and not the time steps t. Hence, in comparing the delay of detection in the time unit, we need to multiply the observed ARL_1 by the corresponding window size n.

4 Conclusions

In this paper, we considered the problem of detecting changes in stationarity in process generating streams of attributed graphs. We adopted a method derived from the methodology proposed in [11] and extended here its experimental analysis. The method operates by embedding the graphs into a vector space by means of the dissimilarity representation. Subsequently, the analysis is attained by a CUSUM-like change detection test on the associated multidimensional stream of all the dissimilarity representation.

The dissimilarity representation grounds on a prototype set and a dissimilarity measure which assesses the mutual distance between observed graphs and prototype graphs. The particular graph dissimilarity measure could be general or specialized for the application at hand. Secondly, the number of prototypes defines the dimension of the embedding space, hence affecting the fidelity of the representation. The change detection procedure receives disjoint windows as inputs and estimates confidence intervals from them. In this way, the size of the window has the twofold role of improving the estimation and attenuate the influence of outliers. As a drawback, increasing the number of prototypes requires more training data, and larger windows produce delays in the detections.

We created *ad hoc* datasets for investigating the effect of two important parameters on our method: the number of prototypes and the size of the window. Experimental results suggest suitable criteria to be used for tuning the method. It is worth mentioning that also the behavior of the dissimilarity measure between graphs is affected by parameters. Here, we used fixed configurations for such parameters. In future works, we will study suitable automatic tuning procedures in order to optimize parameters affecting also the matching procedure taken into account, since it is well-known to significantly affect the overall performance.

Our conclusion is that the proposed method is capable to detect changes in stationarity in processes generating graphs with possibly rich vertex/edge attributes and topologies of variable size, hence stressing its potential applicability in multiple application domains.

Acknowledgements

This research is funded by the Swiss National Science Foundation project 200021_172671: "ALPSFORT: A Learning graPh-baSed framework FOr cybeR-physical sysTems".

References

[1] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, May 2013.

[2] P. Holme, "Modern temporal network theory: A colloquium," *The European Physical Journal B*, vol. 88, no. 9, pp. 1–30, 2015.

[3] B. J. Jain, "On the geometry of graph spaces," *Discrete Applied Mathematics*, vol. 214, pp. 126–144, 2016.

[4] D. M. Hawkins, P. Qiu, and C. W. Kang, "The changepoint model for statistical process control," *Journal of Quality Technology*, vol. 35, no. 4, p. 355, 2003.

[5] M. Basseville and I. V. Nikiforov, *Detection of Abrupt Changes: Theory and Application.* Englewood Cliffs, NJ, USA: Prentice Hall, 1993, vol. 104.

[6] I. Barnett and J.-P. Onnela, "Change point detection in correlation networks," *Scientific Reports*, vol. 6, 2016.

[7] L. Peel and A. Clauset, "Detecting change points in the large-scale structure of evolving networks." in *AAAI*, 2015, pp. 2914–2920.

[8] S. Ranshous, S. Shen, D. Koutra, S. Harenberg, C. Faloutsos, and N. F. Samatova, "Anomaly detection in dynamic networks: A survey," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 7, no. 3, pp. 223–247, 2015.

[9] L. Livi and A. Rizzi, "The graph matching problem," *Pattern Analysis and Applications*, vol. 16, no. 3, pp. 253–283, 2013.

[10] A. Fischer, K. Riesen, and H. Bunke, "Improved quadratic time approximation of graph edit distance by combining Hausdorff matching and greedy assignment," *Pattern Recognition Letters*, vol. 87, pp. 55–62, 2017.

[11] D. Zambon, C. Alippi, and L. Livi, "Concept drift and anomaly detection in graph streams," *arXiv preprint arXiv:1706.06941*, 2017.

[12] R. P. W. Duin and E. Pękalska, "Non-Euclidean dissimilarities: causes and informativeness," in *Proceedings of the 2010 joint IAPR international conference on Structural, syntactic, and statistical pattern recognition.* Springer-Verlag, 2010, pp. 324–333.

[13] E. Pękalska, R. P. W. Duin, and P. Paclík, "Prototype selection for dissimilarity-based classifiers," *Pattern Recognition*, vol. 39, no. 2, pp. 189–208, 2006.

[14] D. Zambon, L. Livi, and C. Alippi, "CDG: (C)hange (D)etection on (G)raph Streams." http://www.inf.usi.ch/phd/zambon/#cdg, 2017, [Online; accessed 25-September-2017].

[15] L. Livi, A. Rizzi, and A. Sadeghian, "Optimized dissimilarity space embedding for labeled graphs," *Information Sciences*, vol. 266, pp. 47–64, 2014.

[16] K. Riesen, S. Emmenegger, and H. Bunke, "A novel software toolkit for graph edit distance computation," in *International Workshop on Graph-Based Representations in Pattern Recognition*. Springer, 2013, pp. 142–151.